Course Project - Air Quality Index for different states across India

READING DATASET

df<-read.csv("C:/Users/RNT/OneDrive/Documents/dsRlab/citywise\_pollutants\_data.csv")  
head(df)

## country state city  
## 1 India Andhra\_Pradesh Amaravati  
## 2 India Andhra\_Pradesh Anantapur  
## 3 India Andhra\_Pradesh Kadapa  
## 4 India Andhra\_Pradesh Rajamahendravaram  
## 5 India Andhra\_Pradesh Rajamahendravaram  
## 6 India Andhra\_Pradesh Rajamahendravaram  
## station last\_update latitude  
## 1 Secretariat, Amaravati - APPCB 2/3/2024 15:00 16.51508  
## 2 Gulzarpet, Anantapur - APPCB 2/3/2024 15:00 14.67589  
## 3 Yerramukkapalli, Kadapa - APPCB 2/3/2024 15:00 14.46505  
## 4 Anand Kala Kshetram, Rajamahendravaram - APPCB 2/3/2024 15:00 16.98729  
## 5 Anand Kala Kshetram, Rajamahendravaram - APPCB 2/3/2024 15:00 16.98729  
## 6 Anand Kala Kshetram, Rajamahendravaram - APPCB 2/3/2024 15:00 16.98729  
## longitude pollutant\_id pollutant\_min pollutant\_max pollutant\_avg  
## 1 80.51817 OZONE 1 45 30  
## 2 77.59303 CO 24 61 40  
## 3 78.82419 SO2 2 2 2  
## 4 81.73632 NH3 4 6 4  
## 5 81.73632 SO2 11 13 12  
## 6 81.73632 CO 25 54 35

nrow(df)

## [1] 3336

ncol(df)

## [1] 11

column\_names<-names(df)  
column\_names

## [1] "country" "state" "city" "station"   
## [5] "last\_update" "latitude" "longitude" "pollutant\_id"   
## [9] "pollutant\_min" "pollutant\_max" "pollutant\_avg"

STRUCTURE OF DATAFRAME

str(df)

## 'data.frame': 3336 obs. of 11 variables:  
## $ country : chr "India" "India" "India" "India" ...  
## $ state : chr "Andhra\_Pradesh" "Andhra\_Pradesh" "Andhra\_Pradesh" "Andhra\_Pradesh" ...  
## $ city : chr "Amaravati" "Anantapur" "Kadapa" "Rajamahendravaram" ...  
## $ station : chr "Secretariat, Amaravati - APPCB" "Gulzarpet, Anantapur - APPCB" "Yerramukkapalli, Kadapa - APPCB" "Anand Kala Kshetram, Rajamahendravaram - APPCB" ...  
## $ last\_update : chr "2/3/2024 15:00" "2/3/2024 15:00" "2/3/2024 15:00" "2/3/2024 15:00" ...  
## $ latitude : num 16.5 14.7 14.5 17 17 ...  
## $ longitude : num 80.5 77.6 78.8 81.7 81.7 ...  
## $ pollutant\_id : chr "OZONE" "CO" "SO2" "NH3" ...  
## $ pollutant\_min: int 1 24 2 4 11 25 9 4 54 1 ...  
## $ pollutant\_max: int 45 61 2 6 13 54 16 43 92 70 ...  
## $ pollutant\_avg: int 30 40 2 4 12 35 13 24 74 34 ...

SUMMARY OF DATAFRAME

summary(df)

## country state city station   
## Length:3336 Length:3336 Length:3336 Length:3336   
## Class :character Class :character Class :character Class :character   
## Mode :character Mode :character Mode :character Mode :character   
##   
##   
##   
##   
## last\_update latitude longitude pollutant\_id   
## Length:3336 Min. : 8.515 Min. :70.91 Length:3336   
## Class :character 1st Qu.:19.058 1st Qu.:75.56 Class :character   
## Mode :character Median :23.228 Median :77.30 Mode :character   
## Mean :22.673 Mean :78.54   
## 3rd Qu.:27.194 3rd Qu.:80.33   
## Max. :34.066 Max. :94.64   
##   
## pollutant\_min pollutant\_max pollutant\_avg   
## Min. : 1.00 Min. : 1.00 Min. : 1.00   
## 1st Qu.: 6.00 1st Qu.: 20.00 1st Qu.: 12.00   
## Median : 15.00 Median : 52.00 Median : 30.00   
## Mean : 22.32 Mean : 79.03 Mean : 42.86   
## 3rd Qu.: 32.00 3rd Qu.:106.00 3rd Qu.: 61.00   
## Max. :149.00 Max. :500.00 Max. :325.00   
## NA's :180 NA's :180 NA's :180

CHECKING THE NULL VALUES (BEFORE DATA CLEANING)

colSums(is.na(df))

## country state city station last\_update   
## 0 0 0 0 0   
## latitude longitude pollutant\_id pollutant\_min pollutant\_max   
## 0 0 0 180 180   
## pollutant\_avg   
## 180

#null vlues in pollutant\_min, pollutant\_max, pollutant\_avg are 180 each i.e 540 in total  
data<-df  
distinct\_pollutants <- unique(data$pollutant\_id)  
distinct\_pollutants

## [1] "OZONE" "CO" "SO2" "NH3" "PM2.5" "NO2" "PM10"

library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

data <- data %>%  
 group\_by(pollutant\_id) %>%  
 mutate(pollutant\_min = ifelse(is.na(pollutant\_min), mean(pollutant\_min, na.rm = TRUE), pollutant\_min),  
 pollutant\_max = ifelse(is.na(pollutant\_max), mean(pollutant\_max, na.rm = TRUE), pollutant\_max),  
 pollutant\_avg = ifelse(is.na(pollutant\_avg), mean(pollutant\_avg, na.rm = TRUE), pollutant\_avg))  
View(data)  
View(df)

CHECKING THE NULL VALUES (AFTER DATA CLEANING)

#colSums(is.na(data))  
na\_count <- colSums(is.na(data[, c("pollutant\_min", "pollutant\_max", "pollutant\_avg")]))  
na\_count

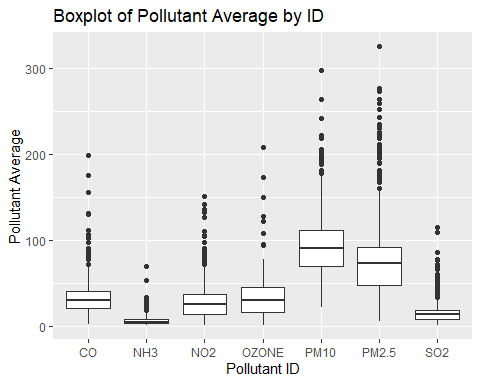
## pollutant\_min pollutant\_max pollutant\_avg   
## 0 0 0

Group the data by pollutant\_id and calculate descriptive statistics

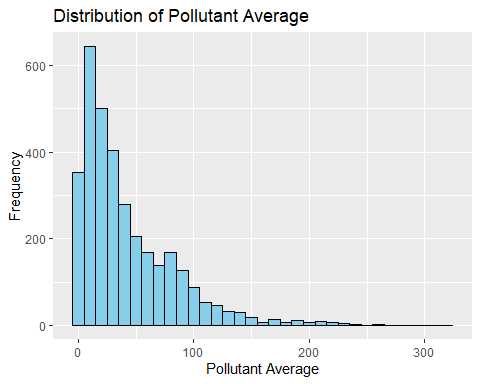
library(dplyr)  
  
pollutant\_statistics <- data %>%  
 group\_by(pollutant\_id) %>%  
 summarize(  
 mean = mean(pollutant\_avg, na.rm = TRUE),  
 median = median(pollutant\_avg, na.rm = TRUE),  
 mode = as.numeric(names(table(pollutant\_avg))[which.max(table(pollutant\_avg))]),  
 std\_dev = sd(pollutant\_avg, na.rm = TRUE),  
 variance = var(pollutant\_avg, na.rm = TRUE)  
 )  
  
# Print the statistics  
print(pollutant\_statistics)

## # A tibble: 7 × 6  
## pollutant\_id mean median mode std\_dev variance  
## <chr> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 CO 35.4 30 18 23.9 569.   
## 2 NH3 6.30 5 2 6.12 37.5  
## 3 NO2 29.7 25.5 29.7 22.7 515.   
## 4 OZONE 32.5 30 6 23.0 530.   
## 5 PM10 94.6 91 94.6 38.8 1507.   
## 6 PM2.5 81.5 73 81.5 49.3 2427.   
## 7 SO2 16.5 14 16.5 14.2 201.

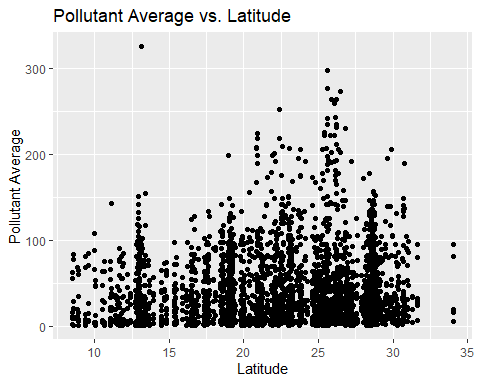
library(ggplot2)  
ggplot(data, aes(x = pollutant\_id, y = pollutant\_avg)) +  
 geom\_boxplot() +  
 labs(x = "Pollutant ID", y = "Pollutant Average", title = "Boxplot of Pollutant Average by ID")



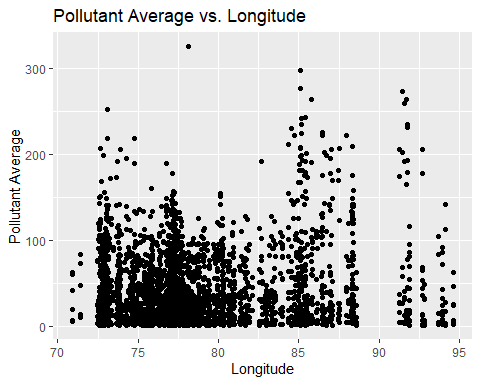
ggplot(data, aes(x = pollutant\_avg)) +  
 geom\_histogram(binwidth = 10, fill = "skyblue", color = "black") +  
 labs(x = "Pollutant Average", y = "Frequency", title = "Distribution of Pollutant Average")



ggplot(data, aes(x = latitude, y = pollutant\_avg)) +  
 geom\_point() +  
 labs(x = "Latitude", y = "Pollutant Average", title = "Pollutant Average vs. Latitude")



ggplot(data, aes(x = longitude, y = pollutant\_avg)) +  
 geom\_point() +  
 labs(x = "Longitude", y = "Pollutant Average", title = "Pollutant Average vs. Longitude")



correlation\_matrix <- cor(data[, c("pollutant\_avg", "latitude", "longitude")], use = "pairwise.complete.obs")  
  
# Printing the correlation matrix  
print(correlation\_matrix)

## pollutant\_avg latitude longitude  
## pollutant\_avg 1.00000000 0.07445768 0.1168606  
## latitude 0.07445768 1.00000000 0.1348671  
## longitude 0.11686057 0.13486709 1.0000000

Calculating AQI for every category of pollutant based on Indian standard

# PM10  
calculate\_AQI\_PM10\_indian <- function(pollutant\_concentration) {  
 breakpoints <- c(0, 50, 100, 250, 350, 430, 430)  
 AQI\_range <- c(0, 50, 100, 200, 300, 400, 500)  
   
 index <- findInterval(pollutant\_concentration, breakpoints)  
 AQI <- ((AQI\_range[index + 1] - AQI\_range[index]) / (breakpoints[index + 1] - breakpoints[index])) \* (pollutant\_concentration - breakpoints[index]) + AQI\_range[index]  
   
 return(AQI)  
}  
  
# PM2.5  
calculate\_AQI\_PM25\_indian <- function(pollutant\_concentration) {  
 breakpoints <- c(0, 30, 60, 90, 120, 250, 430)  
 AQI\_range <- c(0, 50, 100, 200, 300, 400, 500)  
   
 index <- findInterval(pollutant\_concentration, breakpoints)  
 AQI <- ((AQI\_range[index + 1] - AQI\_range[index]) / (breakpoints[index + 1] - breakpoints[index])) \* (pollutant\_concentration - breakpoints[index]) + AQI\_range[index]  
   
 return(AQI)  
}  
  
# NO2  
calculate\_AQI\_NO2\_indian <- function(pollutant\_concentration) {  
 breakpoints <- c(0, 40, 80, 180, 280, 400, 400)  
 AQI\_range <- c(0, 50, 100, 200, 300, 400, 500)  
   
 index <- findInterval(pollutant\_concentration, breakpoints)  
 AQI <- ((AQI\_range[index + 1] - AQI\_range[index]) / (breakpoints[index + 1] - breakpoints[index])) \* (pollutant\_concentration - breakpoints[index]) + AQI\_range[index]  
   
 return(AQI)  
}  
  
# O3  
calculate\_AQI\_O3\_indian <- function(pollutant\_concentration) {  
 breakpoints <- c(0, 50, 100, 168, 208, 748, 748)  
 AQI\_range <- c(0, 50, 100, 200, 300, 400, 500)  
   
 index <- findInterval(pollutant\_concentration, breakpoints)  
 AQI <- ((AQI\_range[index + 1] - AQI\_range[index]) / (breakpoints[index + 1] - breakpoints[index])) \* (pollutant\_concentration - breakpoints[index]) + AQI\_range[index]  
   
 return(AQI)  
}  
  
# CO  
calculate\_AQI\_CO\_indian <- function(pollutant\_concentration) {  
 breakpoints <- c(0, 200, 400, 800, 1200, 1800, 1800)  
 AQI\_range <- c(0, 50, 100, 200, 300, 400, 500)  
   
 index <- findInterval(pollutant\_concentration, breakpoints)  
 AQI <- ((AQI\_range[index + 1] - AQI\_range[index]) / (breakpoints[index + 1] - breakpoints[index])) \* (pollutant\_concentration - breakpoints[index]) + AQI\_range[index]  
   
 return(AQI)  
}  
  
# SO2  
calculate\_AQI\_SO2\_indian <- function(pollutant\_concentration) {  
 breakpoints <- c(0, 40, 80, 380, 800, 1600, 1600)  
 AQI\_range <- c(0, 50, 100, 200, 300, 400, 500)  
   
 index <- findInterval(pollutant\_concentration, breakpoints)  
 AQI <- ((AQI\_range[index + 1] - AQI\_range[index]) / (breakpoints[index + 1] - breakpoints[index])) \* (pollutant\_concentration - breakpoints[index]) + AQI\_range[index]  
   
 return(AQI)  
}  
  
# NH3  
calculate\_AQI\_NH3\_indian <- function(pollutant\_concentration) {  
 breakpoints <- c(0, 200, 400, 800, 1200, 1800, 1800)  
 AQI\_range <- c(0, 50, 100, 200, 300, 400, 500)  
   
 index <- findInterval(pollutant\_concentration, breakpoints)  
 AQI <- ((AQI\_range[index + 1] - AQI\_range[index]) / (breakpoints[index + 1] - breakpoints[index])) \* (pollutant\_concentration - breakpoints[index]) + AQI\_range[index]  
   
 return(AQI)  
}  
  
# Calculate AQI based on pollutant ID  
calculate\_AQI\_indian <- function(pollutant\_concentration, pollutant\_id) {  
 if (pollutant\_id == "PM10") {  
 return(calculate\_AQI\_PM10\_indian(pollutant\_concentration))  
 } else if (pollutant\_id == "PM2.5") {  
 return(calculate\_AQI\_PM25\_indian(pollutant\_concentration))  
 } else if (pollutant\_id == "NO2") {  
 return(calculate\_AQI\_NO2\_indian(pollutant\_concentration))  
 } else if (pollutant\_id == "O3") {  
 return(calculate\_AQI\_O3\_indian(pollutant\_concentration))  
 } else if (pollutant\_id == "CO") {  
 return(calculate\_AQI\_CO\_indian(pollutant\_concentration))  
 } else if (pollutant\_id == "SO2") {  
 return(calculate\_AQI\_SO2\_indian(pollutant\_concentration))  
 } else if (pollutant\_id == "NH3") {  
 return(calculate\_AQI\_NH3\_indian(pollutant\_concentration))  
 } else {  
 return(NA) # For other pollutants, return NA (not applicable)  
 }  
}  
  
  
# Group data by state and calculate AQI for each pollutant for each state  
AQI\_results\_indian <- data %>%  
 group\_by(state) %>%  
 summarize(  
 AQI\_PM10 = max(calculate\_AQI\_indian(pollutant\_min, "PM10")),  
 AQI\_PM25 = max(calculate\_AQI\_indian(pollutant\_min, "PM2.5")),  
 AQI\_NO2 = max(calculate\_AQI\_indian(pollutant\_min, "NO2")),  
 AQI\_O3 = max(calculate\_AQI\_indian(pollutant\_min, "O3")),  
 AQI\_CO = max(calculate\_AQI\_indian(pollutant\_min, "CO")),  
 AQI\_SO2 = max(calculate\_AQI\_indian(pollutant\_min, "SO2")),  
 AQI\_NH3 = max(calculate\_AQI\_indian(pollutant\_min, "NH3")),  
 AQI\_Pb = max(calculate\_AQI\_indian(pollutant\_min, "Pb"))  
 )  
  
# Display the AQI results  
print(AQI\_results\_indian)

## # A tibble: 31 × 9  
## state AQI\_PM10 AQI\_PM25 AQI\_NO2 AQI\_O3 AQI\_CO AQI\_SO2 AQI\_NH3 AQI\_Pb  
## <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <int>  
## 1 Andhra\_Pradesh 83 177. 103 83 20.8 101 20.8 NA  
## 2 Arunachal\_Pra… 37 61.7 46.2 37 9.25 46.2 9.25 NA  
## 3 Assam 127. 315. 160 159. 35 120 35 NA  
## 4 Bihar 126 315. 159 157. 34.8 120. 34.8 NA  
## 5 Chandigarh 85 183. 105 85 21.2 102. 21.2 NA  
## 6 Chhattisgarh 81 170 101 81 20.2 100. 20.2 NA  
## 7 Delhi 115. 302. 142 132. 30.5 114 30.5 NA  
## 8 Gujarat 133. 322. 169 172. 37.2 123 37.2 NA  
## 9 Haryana 103. 250 125 107. 26.2 108. 26.2 NA  
## 10 Himachal Prad… 32 53.3 40 32 8 40 8 NA  
## # ℹ 21 more rows

aqi\_categorywise<-AQI\_results\_indian  
View((aqi\_categorywise))

nrow(aqi\_categorywise)

## [1] 31

distinct\_states<-unique(aqi\_categorywise$state)  
distinct\_states

## [1] "Andhra\_Pradesh" "Arunachal\_Pradesh" "Assam"   
## [4] "Bihar" "Chandigarh" "Chhattisgarh"   
## [7] "Delhi" "Gujarat" "Haryana"   
## [10] "Himachal Pradesh" "Jammu\_and\_Kashmir" "Jharkhand"   
## [13] "Karnataka" "Kerala" "Madhya Pradesh"   
## [16] "Maharashtra" "Manipur" "Meghalaya"   
## [19] "Mizoram" "Nagaland" "Odisha"   
## [22] "Puducherry" "Punjab" "Rajasthan"   
## [25] "Sikkim" "TamilNadu" "Telangana"   
## [28] "Tripura" "Uttar\_Pradesh" "Uttarakhand"   
## [31] "West\_Bengal"

aqi\_categorywise <- aqi\_categorywise %>%  
 select(-c(AQI\_Pb))  
head(aqi\_categorywise)

## # A tibble: 6 × 8  
## state AQI\_PM10 AQI\_PM25 AQI\_NO2 AQI\_O3 AQI\_CO AQI\_SO2 AQI\_NH3  
## <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 Andhra\_Pradesh 83 177. 103 83 20.8 101 20.8   
## 2 Arunachal\_Pradesh 37 61.7 46.2 37 9.25 46.2 9.25  
## 3 Assam 127. 315. 160 159. 35 120 35   
## 4 Bihar 126 315. 159 157. 34.8 120. 34.8   
## 5 Chandigarh 85 183. 105 85 21.2 102. 21.2   
## 6 Chhattisgarh 81 170 101 81 20.2 100. 20.2

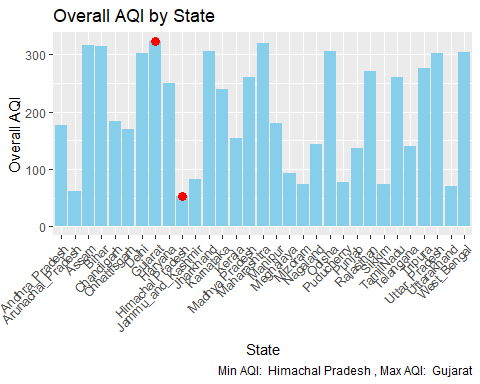
# Calculate the overall AQI for each state by taking the maximum AQI value across all categories  
overall\_AQI <- aqi\_categorywise %>%  
 group\_by(state) %>%  
 summarize(overall\_AQI = max(pmax(AQI\_PM10, AQI\_PM25, AQI\_NO2, AQI\_O3, AQI\_CO, AQI\_SO2, AQI\_NH3), na.rm = TRUE))  
  
# Display the table with states and their respective overall AQI  
print(overall\_AQI)

## # A tibble: 31 × 2  
## state overall\_AQI  
## <chr> <dbl>  
## 1 Andhra\_Pradesh 177.   
## 2 Arunachal\_Pradesh 61.7  
## 3 Assam 315.   
## 4 Bihar 315.   
## 5 Chandigarh 183.   
## 6 Chhattisgarh 170   
## 7 Delhi 302.   
## 8 Gujarat 322.   
## 9 Haryana 250   
## 10 Himachal Pradesh 53.3  
## # ℹ 21 more rows

# Define breakpoints and corresponding categories  
breakpoints <- c(0, 50, 100, 200, 300, 400, 500)  
categories <- c("Good", "Satisfactory", "Moderate", "Poor", "Very Poor", "Severe")  
  
# Classify AQI values into categories  
overall\_AQI$category <- cut(overall\_AQI$overall\_AQI, breakpoints, labels = categories)  
  
# Display the categorized AQI values  
print(overall\_AQI)

## # A tibble: 31 × 3  
## state overall\_AQI category   
## <chr> <dbl> <fct>   
## 1 Andhra\_Pradesh 177. Moderate   
## 2 Arunachal\_Pradesh 61.7 Satisfactory  
## 3 Assam 315. Very Poor   
## 4 Bihar 315. Very Poor   
## 5 Chandigarh 183. Moderate   
## 6 Chhattisgarh 170 Moderate   
## 7 Delhi 302. Very Poor   
## 8 Gujarat 322. Very Poor   
## 9 Haryana 250 Poor   
## 10 Himachal Pradesh 53.3 Satisfactory  
## # ℹ 21 more rows

# Bar chart of overall AQI values for different states(with minimum and maximum values highlighted)  
library(ggplot2)  
  
# Find states with minimum and maximum AQI values  
min\_state <- overall\_AQI[which.min(overall\_AQI$overall\_AQI), "state"]  
max\_state <- overall\_AQI[which.max(overall\_AQI$overall\_AQI), "state"]  
  
# Plot  
ggplot(overall\_AQI, aes(x = state, y = overall\_AQI)) +  
 geom\_bar(stat = "identity", fill = "skyblue") +  
 geom\_point(data = overall\_AQI %>% filter(state %in% c(min\_state, max\_state)),  
 aes(x = state, y = overall\_AQI), color = "red", size = 3) +  
 labs(title = "Overall AQI by State",  
 x = "State",  
 y = "Overall AQI",  
 caption = paste("Min AQI: ", min\_state, ", Max AQI: ", max\_state)) +  
 theme(axis.text.x = element\_text(angle = 45, hjust = 1))



library(caret)

## Warning: package 'caret' was built under R version 4.3.3

## Loading required package: lattice

library(randomForest)

## Warning: package 'randomForest' was built under R version 4.3.3

## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':  
##   
## margin

## The following object is masked from 'package:dplyr':  
##   
## combine

library(glmnet)

## Warning: package 'glmnet' was built under R version 4.3.3

## Loading required package: Matrix

## Loaded glmnet 4.1-8

library(e1071)

## Warning: package 'e1071' was built under R version 4.3.3

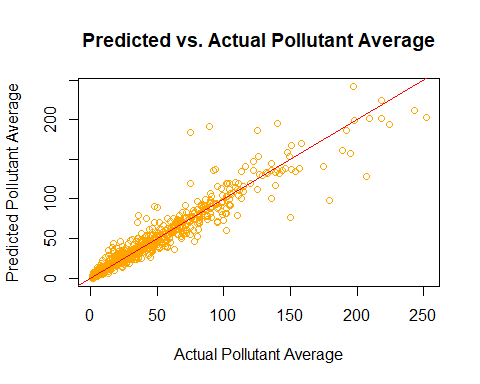
mydata<-data  
  
set.seed(123) # for reproducibility  
train\_indices <- createDataPartition(data$pollutant\_avg, p = 0.75, list = FALSE)  
train\_data <- data[train\_indices, ]  
test\_data <- data[-train\_indices, ]  
head(train\_data)

## # A tibble: 6 × 11  
## # Groups: pollutant\_id [4]  
## country state city station last\_update latitude longitude pollutant\_id  
## <chr> <chr> <chr> <chr> <chr> <dbl> <dbl> <chr>   
## 1 India Andhra\_Prad… Amar… Secret… 2/3/2024 1… 16.5 80.5 OZONE   
## 2 India Andhra\_Prad… Raja… Anand … 2/3/2024 1… 17.0 81.7 NH3   
## 3 India Andhra\_Prad… Tiru… Vaikun… 2/3/2024 1… 13.6 79.4 OZONE   
## 4 India Andhra\_Prad… Vija… Kanuru… 2/3/2024 1… 16.5 80.7 PM2.5   
## 5 India Andhra\_Prad… Vija… Rajiv … 2/3/2024 1… 16.6 80.6 CO   
## 6 India Andhra\_Prad… Visa… GVM Co… 2/3/2024 1… 17.7 83.3 CO   
## # ℹ 3 more variables: pollutant\_min <dbl>, pollutant\_max <dbl>,  
## # pollutant\_avg <dbl>

# LINEAR REGRESSION  
  
model\_lm<-lm(formula=pollutant\_avg~latitude + longitude + pollutant\_min + pollutant\_max,data=train\_data)  
model\_lm

##   
## Call:  
## lm(formula = pollutant\_avg ~ latitude + longitude + pollutant\_min +   
## pollutant\_max, data = train\_data)  
##   
## Coefficients:  
## (Intercept) latitude longitude pollutant\_min pollutant\_max   
## -25.4676 -0.0214 0.3359 0.7847 0.3172

predictions\_lm <- predict(model\_lm, newdata = test\_data)  
plot(test\_data$pollutant\_avg, predictions\_lm,   
 main = "Predicted vs. Actual Pollutant Average",  
 xlab = "Actual Pollutant Average", ylab = "Predicted Pollutant Average",   
 col = "orange")  
abline(0, 1, col = "red")



# Compute evaluation metrics for linear regression model  
MAE\_lm <- mean(abs(predictions\_lm - test\_data$pollutant\_avg))  
MSE\_lm <- mean((predictions\_lm - test\_data$pollutant\_avg)^2)  
RMSE\_lm <- sqrt(MSE\_lm)  
rsquared\_lm <- 1 - (sum((test\_data$pollutant\_avg - predictions\_lm)^2) / sum((test\_data$pollutant\_avg - mean(test\_data$pollutant\_avg))^2))  
  
# Print evaluation metrics  
cat("Mean Absolute Error (MAE):", MAE\_lm, "\n")

## Mean Absolute Error (MAE): 6.194912

cat("Mean Squared Error (MSE):", MSE\_lm, "\n")

## Mean Squared Error (MSE): 140.2078

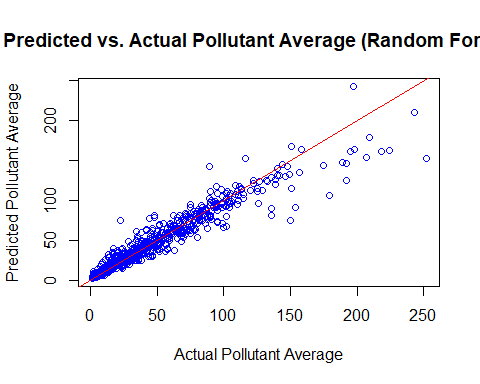
cat("Root Mean Squared Error (RMSE):", RMSE\_lm, "\n")

## Root Mean Squared Error (RMSE): 11.84094

cat("R-squared:", rsquared\_lm, "\n")

## R-squared: 0.9166938

# RANDOM FOREST  
model\_rf <- randomForest(pollutant\_avg ~ latitude + longitude + pollutant\_min + pollutant\_max, data = train\_data)  
predictions\_rf <- predict(model\_rf, newdata = test\_data)  
  
plot(test\_data$pollutant\_avg, predictions\_rf,   
 main = "Predicted vs. Actual Pollutant Average (Random Forest)",  
 xlab = "Actual Pollutant Average", ylab = "Predicted Pollutant Average",   
 col = "blue")  
abline(0, 1, col = "red")



# Compute evaluation metrics for Random Forest model  
MAE\_rf <- mean(abs(predictions\_rf - test\_data$pollutant\_avg))  
MSE\_rf <- mean((predictions\_rf - test\_data$pollutant\_avg)^2)  
RMSE\_rf <- sqrt(MSE\_rf)  
rsquared\_rf <- 1 - (sum((test\_data$pollutant\_avg - predictions\_rf)^2) / sum((test\_data$pollutant\_avg - mean(test\_data$pollutant\_avg))^2))  
  
# Print evaluation metrics for Random Forest model  
cat("Random Forest Model Evaluation Metrics:\n")

## Random Forest Model Evaluation Metrics:

cat("Mean Absolute Error (MAE):", MAE\_rf, "\n")

## Mean Absolute Error (MAE): 6.375536

cat("Mean Squared Error (MSE):", MSE\_rf, "\n")

## Mean Squared Error (MSE): 136.0494

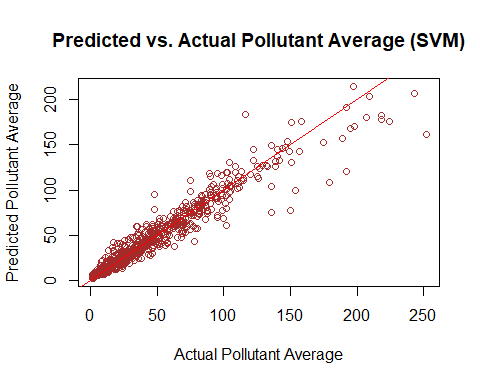
cat("Root Mean Squared Error (RMSE):", RMSE\_rf, "\n")

## Root Mean Squared Error (RMSE): 11.66402

cat("R-squared:", rsquared\_rf, "\n")

## R-squared: 0.9191646

# SUPPORT VECTOR MACHINE  
model\_svm <- svm(pollutant\_avg ~ latitude + longitude + pollutant\_min + pollutant\_max, data = train\_data)  
predictions\_svm <- predict(model\_svm, newdata = test\_data)  
plot(test\_data$pollutant\_avg, predictions\_svm,   
 main = "Predicted vs. Actual Pollutant Average (SVM)",  
 xlab = "Actual Pollutant Average", ylab = "Predicted Pollutant Average",   
 col = "brown")  
abline(0, 1, col = "red")



# Compute evaluation metrics for SVM model  
MAE\_svm <- mean(abs(predictions\_svm - test\_data$pollutant\_avg))  
MSE\_svm <- mean((predictions\_svm - test\_data$pollutant\_avg)^2)  
RMSE\_svm <- sqrt(MSE\_svm)  
rsquared\_svm <- 1 - (sum((test\_data$pollutant\_avg - predictions\_svm)^2) / sum((test\_data$pollutant\_avg - mean(test\_data$pollutant\_avg))^2))  
  
# Print evaluation metrics for SVM model  
cat("SVM Model Evaluation Metrics:\n")

## SVM Model Evaluation Metrics:

cat("Mean Absolute Error (MAE):", MAE\_svm, "\n")

## Mean Absolute Error (MAE): 6.029466

cat("Mean Squared Error (MSE):", MSE\_svm, "\n")

## Mean Squared Error (MSE): 116.1087

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

## Root Mean Squared Error (RMSE): 10.77537

cat("R-squared:", rsquared\_svm, "\n")

## R-squared: 0.9310126

# GRADIENT BOOSTING MACHINE  
library(gbm)

## Warning: package 'gbm' was built under R version 4.3.3

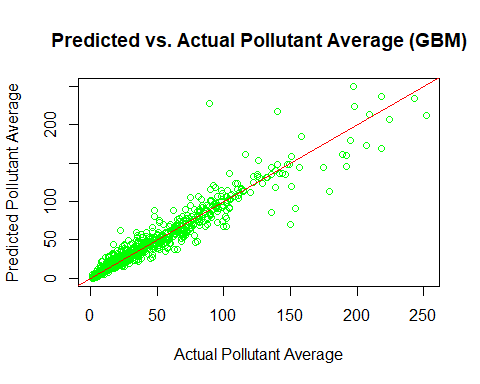
## Loaded gbm 2.1.9

## This version of gbm is no longer under development. Consider transitioning to gbm3, https://github.com/gbm-developers/gbm3

model\_gbm <- gbm(pollutant\_avg ~ latitude + longitude + pollutant\_min + pollutant\_max, data = train\_data, n.trees = 1000, interaction.depth = 4)

## Distribution not specified, assuming gaussian ...

predictions\_gbm <- predict(model\_gbm, newdata = test\_data, n.trees = 1000)  
  
plot(test\_data$pollutant\_avg, predictions\_gbm,   
 main = "Predicted vs. Actual Pollutant Average (GBM)",  
 xlab = "Actual Pollutant Average", ylab = "Predicted Pollutant Average",   
 col = "green")  
abline(0, 1, col = "red")



library(gbm)  
  
# Compute evaluation metrics for GBM model  
MAE\_gbm <- mean(abs(predictions\_gbm - test\_data$pollutant\_avg))  
MSE\_gbm <- mean((predictions\_gbm - test\_data$pollutant\_avg)^2)  
RMSE\_gbm <- sqrt(MSE\_gbm)  
rsquared\_gbm <- 1 - (sum((test\_data$pollutant\_avg - predictions\_gbm)^2) / sum((test\_data$pollutant\_avg - mean(test\_data$pollutant\_avg))^2))  
  
# Print evaluation metrics for GBM model  
cat("Gradient Boosting Machine (GBM) Model Evaluation Metrics:\n")

## Gradient Boosting Machine (GBM) Model Evaluation Metrics:

cat("Mean Absolute Error (MAE):", MAE\_gbm, "\n")

## Mean Absolute Error (MAE): 6.068389

cat("Mean Squared Error (MSE):", MSE\_gbm, "\n")

## Mean Squared Error (MSE): 135.6276

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

## Root Mean Squared Error (RMSE): 11.64592

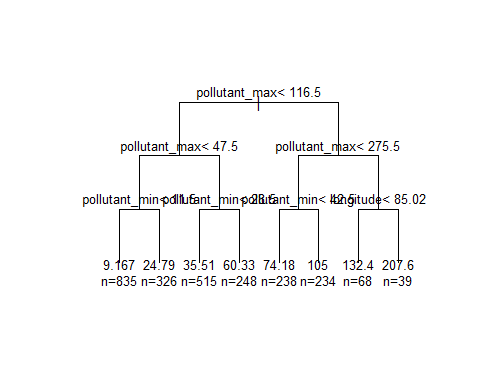
cat("R-squared:", rsquared\_gbm, "\n")

## R-squared: 0.9194152

# DECISION TREES  
library(rpart)

## Warning: package 'rpart' was built under R version 4.3.3

model\_tree <- rpart(pollutant\_avg ~ latitude + longitude + pollutant\_min + pollutant\_max, data = train\_data)  
  
predictions\_tree <- predict(model\_tree, newdata = test\_data)  
  
plot(model\_tree, uniform = TRUE, compress = TRUE, margin = 0.1)  
text(model\_tree, use.n = TRUE, cex = 0.8)



library(rpart)  
  
# Compute evaluation metrics for Decision Tree model  
MAE\_tree <- mean(abs(predictions\_tree - test\_data$pollutant\_avg))  
MSE\_tree <- mean((predictions\_tree - test\_data$pollutant\_avg)^2)  
RMSE\_tree <- sqrt(MSE\_tree)  
rsquared\_tree <- 1 - (sum((test\_data$pollutant\_avg - predictions\_tree)^2) / sum((test\_data$pollutant\_avg - mean(test\_data$pollutant\_avg))^2))  
  
# Print evaluation metrics for Decision Tree model  
cat("Decision Tree Model Evaluation Metrics:\n")

## Decision Tree Model Evaluation Metrics:

cat("Mean Absolute Error (MAE):", MAE\_tree, "\n")

## Mean Absolute Error (MAE): 10.43243

cat("Mean Squared Error (MSE):", MSE\_tree, "\n")

## Mean Squared Error (MSE): 259.8205

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

## Root Mean Squared Error (RMSE): 16.11895

cat("R-squared:", rsquared\_tree, "\n")

## R-squared: 0.8456244

# Define evaluation metrics for all models  
model\_metrics <- data.frame(  
 Model = c("Linear Regression", "Random Forest", "SVM", "GBM", "Decision Tree"),  
 MAE = c(MAE\_lm, MAE\_rf, MAE\_svm, MAE\_gbm, MAE\_tree),  
 MSE = c(MSE\_lm, MSE\_rf, MSE\_svm, MSE\_gbm, MSE\_tree),  
 RMSE = c(RMSE\_lm, RMSE\_rf, RMSE\_svm, RMSE\_gbm, RMSE\_tree),  
 R\_squared = c(rsquared\_lm, rsquared\_rf, rsquared\_svm, rsquared\_gbm, rsquared\_tree)  
)  
  
# Print the model metrics  
print(model\_metrics)

## Model MAE MSE RMSE R\_squared  
## 1 Linear Regression 6.194912 140.2078 11.84094 0.9166938  
## 2 Random Forest 6.375536 136.0494 11.66402 0.9191646  
## 3 SVM 6.029466 116.1087 10.77537 0.9310126  
## 4 GBM 6.068389 135.6276 11.64592 0.9194152  
## 5 Decision Tree 10.432433 259.8205 16.11895 0.8456244

As we can see, the error produced in SVM is the least, thus has the best performance out of all other models