ModelB.32

Group K

2025-10-03

## Load Libraries

library(dplyr)  
library(ggplot2)  
library(tidyr)  
library(reshape2)  
library(corrplot)  
library(here)  
library(caret)  
library(randomForest)  
library(gridExtra)  
library(cluster)  
library(factoextra)  
library(xgboost)

## Data Loading and Initial Exploration

getwd()

## [1] "C:/Users/Tshiamo/OneDrive/Documents/Belgium Campus iTversity/BComp 2025/BIN381/Project/BIN381-Project/scripts/Modelling Scripts"

list.files(recursive = TRUE)

## [1] "Feature-Selection---Member-3.Rmd"   
## [2] "Feature Selection - Member 3.Rmd"   
## [3] "Modelling-Dataset-A-Dean.Rmd"   
## [4] "Modelling group C.Rmd"   
## [5] "Modelling\_Group\_B.docx"   
## [6] "Modelling\_Group\_B.Rmd"   
## [7] "Modelling\_Group\_B.tex"   
## [8] "Modelling\_Group\_B\_files/figure-html/unnamed-chunk-10-1.png"  
## [9] "Modelling\_Group\_B\_files/figure-html/unnamed-chunk-10-2.png"  
## [10] "Modelling\_Group\_B\_files/figure-html/unnamed-chunk-13-1.png"  
## [11] "Modelling\_Group\_B\_files/figure-html/unnamed-chunk-13-2.png"  
## [12] "Modelling\_Group\_B\_files/figure-html/unnamed-chunk-4-1.png"   
## [13] "Modelling\_Group\_B\_files/figure-html/unnamed-chunk-4-2.png"   
## [14] "Modelling\_Group\_B\_files/figure-html/unnamed-chunk-4-3.png"   
## [15] "Modelling\_Group\_B\_files/figure-html/unnamed-chunk-9-1.png"   
## [16] "Modelling\_Group\_B\_files/Modelling\_Group\_B.pdf"   
## [17] "outputsB/visuals/dist\_by\_indicator.png"   
## [18] "outputsB/visuals/value\_vs\_denweighted.png"   
## [19] "rf\_variable\_importance.png"

# Load data  
san\_df <- read.csv(here("merged datasets", "GroupB\_Sanitation\_merged.csv"))  
  
# Initial data structure  
str(san\_df)

## 'data.frame': 142 obs. of 10 variables:  
## $ Dataset : chr "Toilet\_05" "Toilet\_05" "Toilet\_05" "Toilet\_05" ...  
## $ SurveyYear : int 1998 1998 1998 1998 1998 1998 1998 1998 1998 1998 ...  
## $ CharacteristicId : int 1000 1000 1000 1000 1000 1000 1000 1000 1000 1000 ...  
## $ CharacteristicCategory: chr "Total" "Total" "Total" "Total" ...  
## $ CharacteristicLabel : chr "Total" "Total" "Total" "Total" ...  
## $ IndicatorId : chr "WS\_TLET\_H\_IMP" "WS\_TLET\_H\_NIM" "WS\_TLET\_H\_NPT" "WS\_TLET\_H\_NBK" ...  
## $ IndicatorType : chr "I" "I" "I" "I" ...  
## $ Value : num 50.1 38.3 31.2 6 11.6 ...  
## $ DenominatorWeighted : int 12247 12247 12247 12247 12247 12247 12247 NA 12247 52007 ...  
## $ DenominatorUnweighted : int 12247 12247 12247 12247 12247 12247 12247 12247 NA 52465 ...

# Missing value analysis  
missing\_summary <- data.frame(  
 Column = names(san\_df),  
 Missing\_Count = sapply(san\_df, function(x) sum(is.na(x))),  
 Missing\_Percent = round(sapply(san\_df, function(x) sum(is.na(x))/length(x)\*100), 2)  
)  
  
print(missing\_summary)

## Column Missing\_Count Missing\_Percent  
## Dataset Dataset 0 0.00  
## SurveyYear SurveyYear 0 0.00  
## CharacteristicId CharacteristicId 0 0.00  
## CharacteristicCategory CharacteristicCategory 0 0.00  
## CharacteristicLabel CharacteristicLabel 0 0.00  
## IndicatorId IndicatorId 0 0.00  
## IndicatorType IndicatorType 0 0.00  
## Value Value 0 0.00  
## DenominatorWeighted DenominatorWeighted 4 2.82  
## DenominatorUnweighted DenominatorUnweighted 4 2.82

# Unique value analysis  
unique\_summary <- sapply(san\_df, function(x) {  
 vals <- unique(x)  
 paste0(length(vals), " -> ", paste(head(vals, 3), collapse = ", "))  
})  
print(unique\_summary)

## Dataset   
## "2 -> Toilet\_05, Water\_06"   
## SurveyYear   
## "2 -> 1998, 2016"   
## CharacteristicId   
## "1 -> 1000"   
## CharacteristicCategory   
## "1 -> Total"   
## CharacteristicLabel   
## "1 -> Total"   
## IndicatorId   
## "94 -> WS\_TLET\_H\_IMP, WS\_TLET\_H\_NIM, WS\_TLET\_H\_NPT"   
## IndicatorType   
## "5 -> I, S, T"   
## Value   
## "88 -> 50.1, 38.3, 31.2"   
## DenominatorWeighted   
## "5 -> 12247, NA, 52007"   
## DenominatorUnweighted   
## "5 -> 12247, NA, 52465"

## Data Cleaning and Preparation

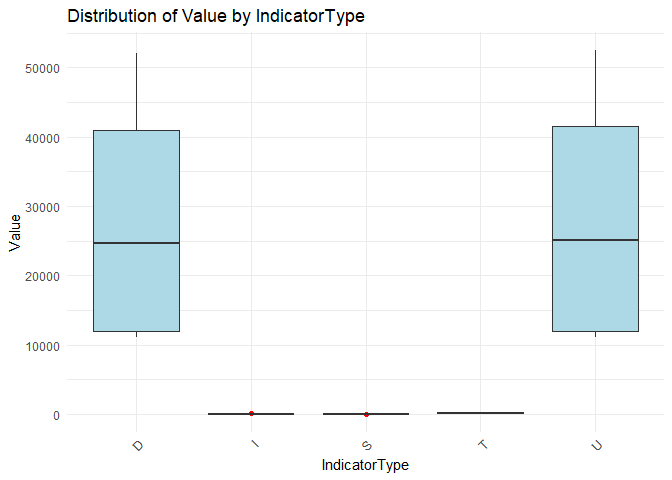
# Filter out NA values from target variable and create clean dataset  
plot\_data <- san\_df %>% filter(!is.na(Value))  
  
# Handle missing values in denominator columns  
plot\_data <- plot\_data %>%  
 mutate(  
 DenominatorWeighted = ifelse(is.na(DenominatorWeighted),  
 median(DenominatorWeighted, na.rm = TRUE),  
 DenominatorWeighted),  
 DenominatorUnweighted = ifelse(is.na(DenominatorUnweighted),  
 median(DenominatorUnweighted, na.rm = TRUE),  
 DenominatorUnweighted)  
 )  
  
cat("Final dataset dimensions:", dim(plot\_data))

## Final dataset dimensions: 142 10

## Exploratory Data Analysis

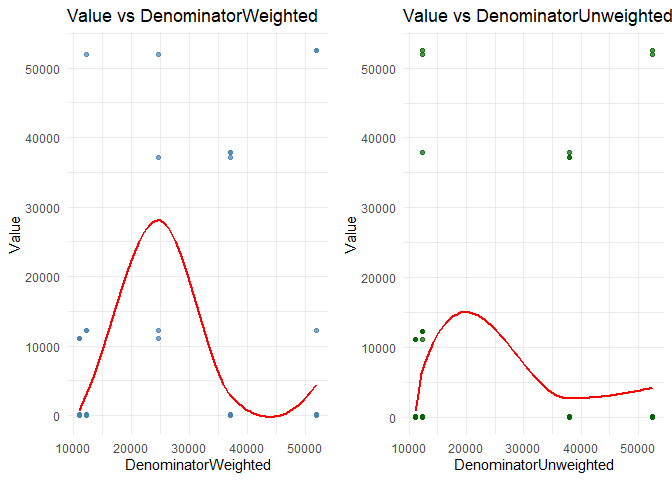
### Distribution of Target Variable by Category

# Determine which categorical variable to use for grouping  
label\_col <- ifelse(length(unique(plot\_data$IndicatorType)) <= 12,  
 'IndicatorType', 'CharacteristicCategory')  
plot\_data[[label\_col]] <- as.factor(plot\_data[[label\_col]])  
  
ggplot(plot\_data, aes\_string(x = label\_col, y = 'Value')) +  
 geom\_boxplot(outlier.colour = "red", fill = "lightblue") +  
 theme\_minimal() +  
 theme(axis.text.x = element\_text(angle = 45, hjust = 1)) +  
 labs(title = paste('Distribution of Value by', label\_col),  
 x = label\_col, y = 'Value')



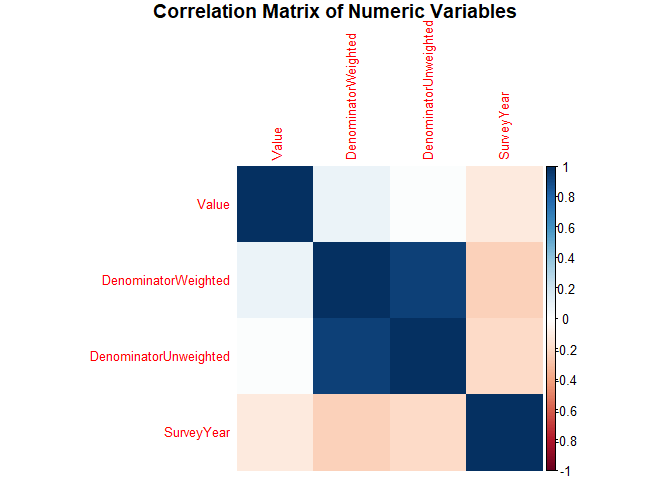
### Relationship with Denominator Variables

# Value vs DenominatorWeighted  
p1 <- plot\_data %>%   
 ggplot(aes(x = DenominatorWeighted, y = Value)) +   
 geom\_point(alpha = 0.7, color = "steelblue") +   
 geom\_smooth(method = 'loess', se = FALSE, color = "red") +   
 theme\_minimal() +  
 labs(title = 'Value vs DenominatorWeighted',   
 x = 'DenominatorWeighted', y = 'Value')  
  
# Value vs DenominatorUnweighted  
p2 <- plot\_data %>%   
 ggplot(aes(x = DenominatorUnweighted, y = Value)) +   
 geom\_point(alpha = 0.7, color = "darkgreen") +   
 geom\_smooth(method = 'loess', se = FALSE, color = "red") +   
 theme\_minimal() +  
 labs(title = 'Value vs DenominatorUnweighted',   
 x = 'DenominatorUnweighted', y = 'Value')  
  
grid.arrange(p1, p2, ncol = 2)



### Correlation Analysis

# Correlation matrix for numeric variables  
num\_cols <- c('Value', 'DenominatorWeighted', 'DenominatorUnweighted', 'SurveyYear')  
cor\_mat <- cor(plot\_data[, num\_cols], use = 'pairwise.complete.obs')  
  
corrplot(cor\_mat, method = 'color', tl.cex = 0.8,   
 title = "Correlation Matrix of Numeric Variables",  
 mar = c(0, 0, 1, 0))



## Feature Engineering and Preprocessing

# Function to drop constant columns  
drop\_constant <- function(df) {  
 keep <- sapply(df, function(x) length(unique(x[!is.na(x)])) > 1)  
 dropped <- names(df)[!keep]  
 if (length(dropped) > 0) {  
 message("Dropped constant columns: ", paste(dropped, collapse = ", "))  
 }  
 return(df[, keep, drop = FALSE])  
}  
  
# Prepare modeling dataset  
mod\_df <- plot\_data %>%  
 mutate(  
 IndicatorType = as.factor(IndicatorType),  
 CharacteristicCategory = as.factor(CharacteristicCategory),  
 # Create new features  
 Weight\_Ratio = DenominatorWeighted / DenominatorUnweighted,  
 Log\_Value = log(Value + 1) # For potential transformation  
 )  
  
# Drop constant columns  
mod\_df <- drop\_constant(mod\_df)  
  
cat("Modeling dataset dimensions:", dim(mod\_df))

## Modeling dataset dimensions: 142 9

# One-hot encode categorical variables  
dummies <- dummyVars(Value ~ ., data = mod\_df)  
X <- predict(dummies, newdata = mod\_df)  
y <- mod\_df$Value  
  
# Train/test split  
set.seed(42)  
train\_idx <- createDataPartition(y, p = 0.8, list = FALSE)  
X\_train <- X[train\_idx, ]; X\_test <- X[-train\_idx, ]  
y\_train <- y[train\_idx]; y\_test <- y[-train\_idx]  
  
cat("Training set size:", length(y\_train))

## Training set size: 114

cat("Test set size:", length(y\_test))

## Test set size: 28

## Model 1: Random Forest

# Train Random Forest model  
set.seed(42)  
rf\_model <- randomForest(x = X\_train, y = y\_train,   
 ntree = 500,   
 importance = TRUE,  
 do.trace = 100)

## | Out-of-bag |  
## Tree | MSE %Var(y) |  
## 100 | 8.109e+06 5.88 |  
## 200 | 7.657e+06 5.55 |  
## 300 | 8.124e+06 5.89 |  
## 400 | 7.84e+06 5.68 |  
## 500 | 7.704e+06 5.58 |

# Predictions  
rf\_pred <- predict(rf\_model, X\_test)  
  
# Performance metrics  
rf\_rmse <- sqrt(mean((y\_test - rf\_pred)^2))  
rf\_mae <- mean(abs(y\_test - rf\_pred))  
rf\_r2 <- cor(y\_test, rf\_pred)^2  
  
cat("Random Forest Performance:\n")

## Random Forest Performance:

cat("RMSE:", round(rf\_rmse, 2), "\n")

## RMSE: 243.58

cat("MAE:", round(rf\_mae, 2), "\n")

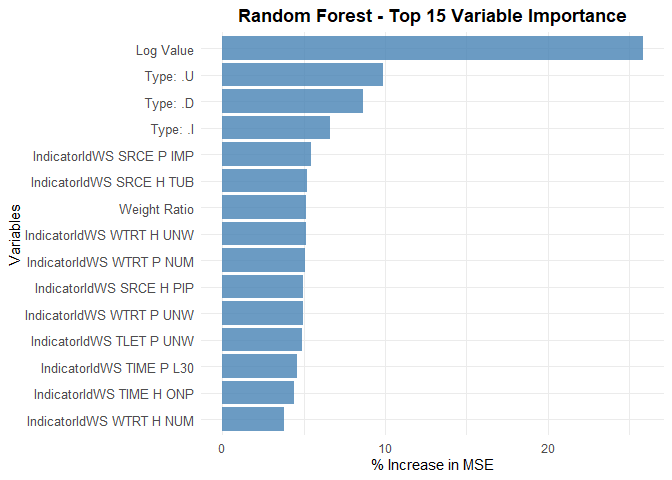
## MAE: 86.78

cat("R^2:", round(rf\_r2, 3), "\n")

## R^2: 0.999

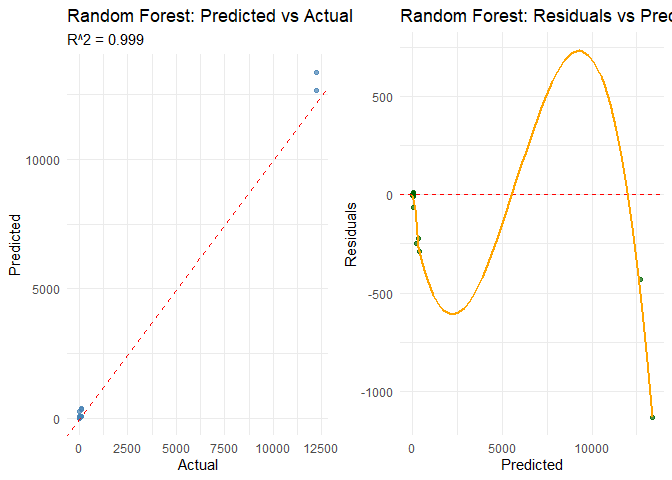
### Random Forest Variable Importance

# Extract and plot variable importance  
importance\_df <- as.data.frame(importance(rf\_model))  
importance\_df$Variable <- rownames(importance\_df)  
  
# Clean variable names for better readability  
importance\_df$Variable <- gsub("\_", " ", importance\_df$Variable)  
importance\_df$Variable <- gsub("IndicatorType", "Type: ", importance\_df$Variable)  
importance\_df$Variable <- gsub("CharacteristicCategory", "Category: ", importance\_df$Variable)  
  
# Plot top 15 most important variables  
importance\_df <- importance\_df[order(-importance\_df$`%IncMSE`), ]  
top\_vars <- head(importance\_df, 15)  
  
ggplot(top\_vars, aes(x = reorder(Variable, `%IncMSE`), y = `%IncMSE`)) +  
 geom\_col(fill = "steelblue", alpha = 0.8) +  
 coord\_flip() +  
 theme\_minimal() +  
 labs(title = "Random Forest - Top 15 Variable Importance",  
 x = "Variables",  
 y = "% Increase in MSE") +  
 theme(plot.title = element\_text(hjust = 0.5, face = "bold"),  
 axis.text.y = element\_text(size = 10))



### Random Forest Diagnostics

# Predicted vs Actual plot  
rf\_plot\_df <- data.frame(Actual = y\_test, Predicted = rf\_pred)  
  
p1 <- ggplot(rf\_plot\_df, aes(x = Actual, y = Predicted)) +  
 geom\_point(alpha = 0.7, color = "steelblue") +  
 geom\_abline(slope = 1, intercept = 0, linetype = "dashed", color = "red") +  
 theme\_minimal() +  
 labs(title = "Random Forest: Predicted vs Actual",  
 subtitle = paste("R^2 =", round(rf\_r2, 3)),  
 x = "Actual", y = "Predicted")  
  
# Residuals plot  
p2 <- ggplot(rf\_plot\_df, aes(x = Predicted, y = Actual - Predicted)) +  
 geom\_point(alpha = 0.7, color = "darkgreen") +  
 geom\_hline(yintercept = 0, linetype = "dashed", color = "red") +  
 geom\_smooth(se = FALSE, color = "orange") +  
 theme\_minimal() +  
 labs(title = "Random Forest: Residuals vs Predicted",  
 x = "Predicted", y = "Residuals")  
  
grid.arrange(p1, p2, ncol = 2)



## Model 2: XGBoost

# Prepare data for XGBoost  
dtrain <- xgb.DMatrix(data = X\_train, label = y\_train)  
dtest <- xgb.DMatrix(data = X\_test, label = y\_test)

# XGBoost parameters with regularization  
params <- list(  
 objective = 'reg:squarederror',  
 eta = 0.1,  
 max\_depth = 6,  
 subsample = 0.8,  
 colsample\_bytree = 0.8,  
 lambda = 1, # L2 regularization  
 alpha = 0.5 # L1 regularization  
)  
  
# Train XGBoost model  
set.seed(42)  
xgb\_mod <- xgboost::xgb.train(  
 params = params,  
 data = dtrain,  
 nrounds = 200,  
 watchlist = list(train = dtrain, test = dtest),  
 verbose = 0,  
 print\_every\_n = 50  
)  
  
# Predictions  
xgb\_pred <- predict(xgb\_mod, dtest)  
  
# Performance metrics  
xgb\_rmse <- sqrt(mean((y\_test - xgb\_pred)^2))  
xgb\_mae <- mean(abs(y\_test - xgb\_pred))  
xgb\_r2 <- cor(y\_test, xgb\_pred)^2  
  
cat("XGBoost Performance:\n")

## XGBoost Performance:

cat("RMSE:", round(xgb\_rmse, 2), "\n")

## RMSE: 202.28

cat("MAE:", round(xgb\_mae, 2), "\n")

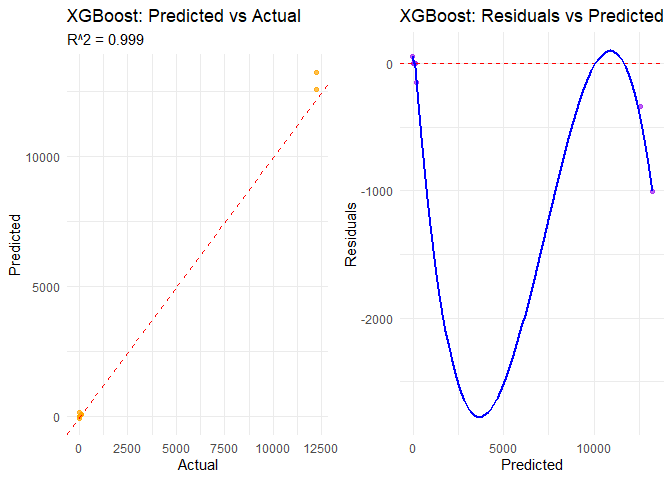
## MAE: 55.87

cat("R^2:", round(xgb\_r2, 3), "\n")

## R^2: 0.999

### XGBoost Diagnostics

# Predicted vs Actual plot  
xgb\_plot\_df <- data.frame(Actual = y\_test, Predicted = xgb\_pred)  
  
p1 <- ggplot(xgb\_plot\_df, aes(x = Actual, y = Predicted)) +  
 geom\_point(alpha = 0.7, color = "orange") +  
 geom\_abline(slope = 1, intercept = 0, linetype = "dashed", color = "red") +  
 theme\_minimal() +  
 labs(title = "XGBoost: Predicted vs Actual",  
 subtitle = paste("R^2 =", round(xgb\_r2, 3)),  
 x = "Actual", y = "Predicted")  
  
# Residuals plot  
p2 <- ggplot(xgb\_plot\_df, aes(x = Predicted, y = Actual - Predicted)) +  
 geom\_point(alpha = 0.7, color = "purple") +  
 geom\_hline(yintercept = 0, linetype = "dashed", color = "red") +  
 geom\_smooth(se = FALSE, color = "blue") +  
 theme\_minimal() +  
 labs(title = "XGBoost: Residuals vs Predicted",  
 x = "Predicted", y = "Residuals")  
  
grid.arrange(p1, p2, ncol = 2)

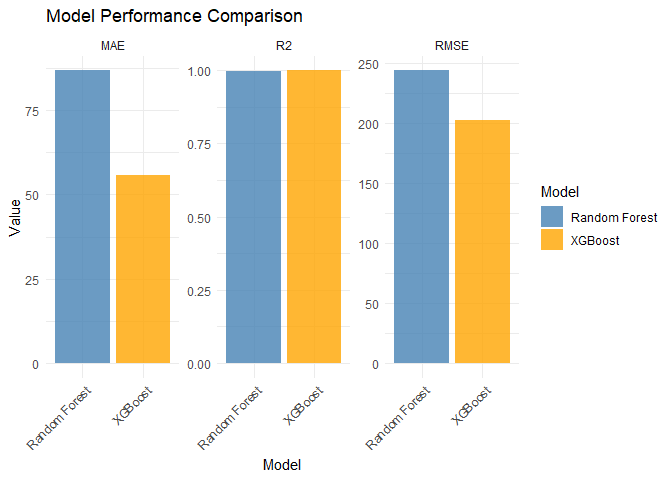


## Model Comparison

# Create comparison table  
model\_comparison <- data.frame(  
 Model = c("Random Forest", "XGBoost"),  
 RMSE = c(rf\_rmse, xgb\_rmse),  
 MAE = c(rf\_mae, xgb\_mae),  
 R2 = c(rf\_r2, xgb\_r2)  
)  
  
print(model\_comparison)

## Model RMSE MAE R2  
## 1 Random Forest 243.5839 86.78314 0.9986748  
## 2 XGBoost 202.2786 55.86616 0.9992020

# Visual comparison  
comparison\_plot <- model\_comparison %>%  
 pivot\_longer(cols = c(RMSE, MAE, R2), names\_to = "Metric", values\_to = "Value") %>%  
 ggplot(aes(x = Model, y = Value, fill = Model)) +  
 geom\_col(alpha = 0.8) +  
 facet\_wrap(~Metric, scales = "free\_y") +  
 theme\_minimal() +  
 theme(axis.text.x = element\_text(angle = 45, hjust = 1)) +  
 labs(title = "Model Performance Comparison",  
 y = "Value") +  
 scale\_fill\_manual(values = c("Random Forest" = "steelblue", "XGBoost" = "orange"))  
  
print(comparison\_plot)



## Final Model Selection and Analysis

# Select best model based on R²  
if (rf\_r2 >= xgb\_r2) {  
 best\_model <- "Random Forest"  
 best\_pred <- rf\_pred  
 best\_rmse <- rf\_rmse  
 best\_mae <- rf\_mae  
 best\_r2 <- rf\_r2  
} else {  
 best\_model <- "XGBoost"  
 best\_pred <- xgb\_pred  
 best\_rmse <- xgb\_rmse  
 best\_mae <- xgb\_mae  
 best\_r2 <- xgb\_r2  
}  
  
cat("Selected Best Model:", best\_model, "\n")

## Selected Best Model: XGBoost

cat("Final Performance Metrics:\n")

## Final Performance Metrics:

cat("RMSE:", round(best\_rmse, 2), "\n")

## RMSE: 202.28

cat("MAE:", round(best\_mae, 2), "\n")

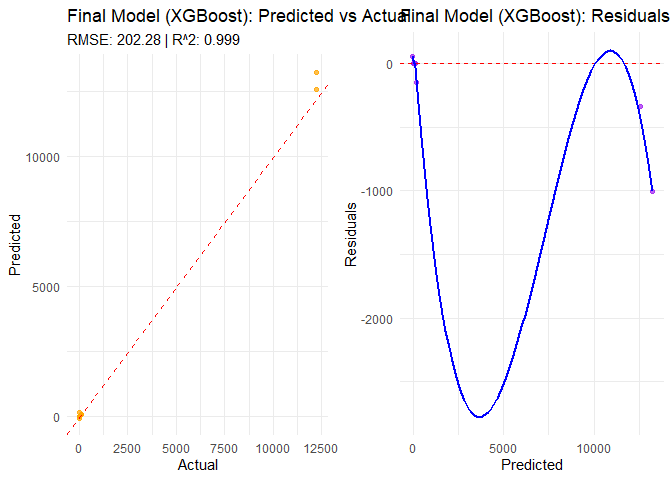
## MAE: 55.87

cat("R^2:", round(best\_r2, 3), "\n")

## R^2: 0.999

### Final Model Diagnostics

final\_plot\_df <- data.frame(Actual = y\_test, Predicted = best\_pred)  
  
p1 <- ggplot(final\_plot\_df, aes(x = Actual, y = Predicted)) +  
 geom\_point(alpha = 0.7, color = ifelse(best\_model == "Random Forest", "steelblue", "orange")) +  
 geom\_abline(slope = 1, intercept = 0, linetype = "dashed", color = "red") +  
 theme\_minimal() +  
 labs(title = paste("Final Model (", best\_model, "): Predicted vs Actual", sep = ""),  
 subtitle = paste("RMSE:", round(best\_rmse, 2), "| R^2:", round(best\_r2, 3)),  
 x = "Actual", y = "Predicted")  
  
p2 <- ggplot(final\_plot\_df, aes(x = Predicted, y = Actual - Predicted)) +  
 geom\_point(alpha = 0.7, color = ifelse(best\_model == "Random Forest", "darkgreen", "purple")) +  
 geom\_hline(yintercept = 0, linetype = "dashed", color = "red") +  
 geom\_smooth(se = FALSE, color = "blue") +  
 theme\_minimal() +  
 labs(title = paste("Final Model (", best\_model, "): Residuals", sep = ""),  
 x = "Predicted", y = "Residuals")  
  
grid.arrange(p1, p2, ncol = 2)



## Conclusion

# Summary statistics  
cat("Dataset Summary:\n")

## Dataset Summary:

cat("Original observations:", nrow(san\_df), "\n")

## Original observations: 142

cat("After cleaning:", nrow(plot\_data), "\n")

## After cleaning: 142

cat("Training samples:", length(y\_train), "\n")

## Training samples: 114

cat("Test samples:", length(y\_test), "\n")

## Test samples: 28

cat("Number of features:", ncol(X), "\n")

## Number of features: 106

cat("\nKey Findings:\n")

##   
## Key Findings:

cat("- Best performing model:", best\_model, "\n")

## - Best performing model: XGBoost

cat("- Final R^2 on test set:", round(best\_r2, 3), "\n")

## - Final R^2 on test set: 0.999

cat("- Model error (RMSE):", round(best\_rmse, 2), "\n")

## - Model error (RMSE): 202.28