Alarm Management Analysis Using Machine Learning

Team 5

2025-05-04

Introduction

This report presents an end-to-end data analytics process to manage and classify industrial alarms, focusing on chattering behavior and predicting alarm durations using machine learning models including Logistic Regression, Decision Trees, and XGBoost.

Load Required Libraries

```
library(readxl)
library(tidyverse)
library(knitr)
library(RColorBrewer)
library(gridExtra)
library(grid)
library(caret)
library(rpart)
library(xgboost)
library(pROC)
library(e1071)
library(Metrics)
```

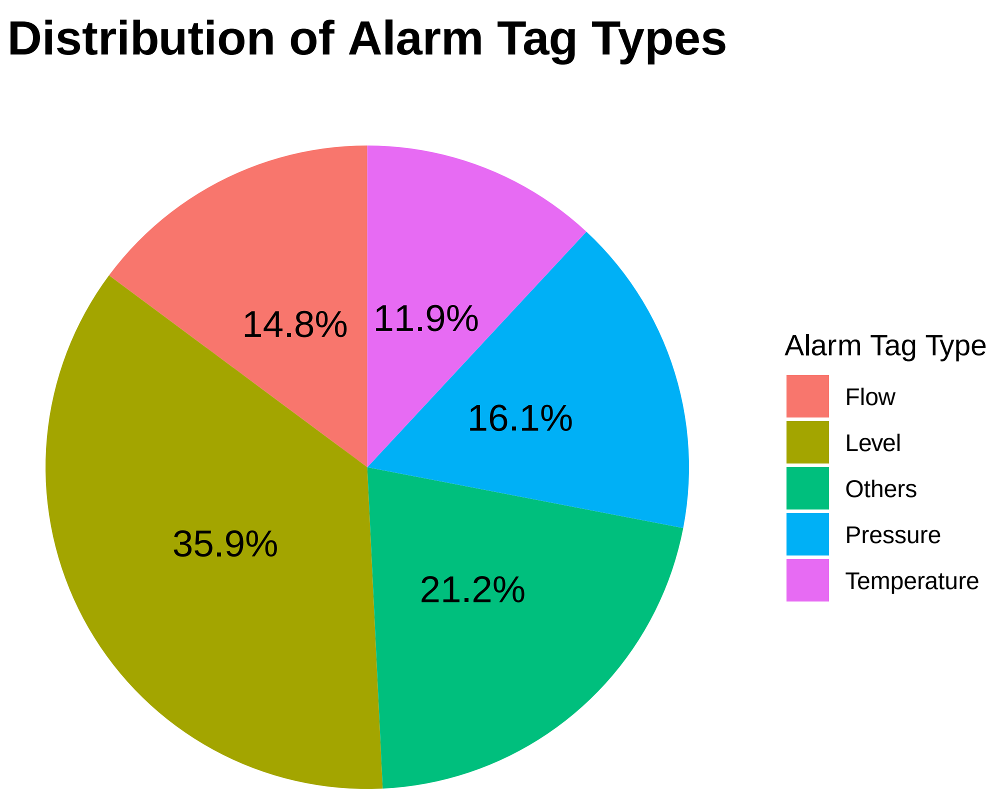
Load and Preview Dataset

```
data <- read_excel("IM009B-XLS-ENG.xlsx")
head(data)
```

```
## # A tibble: 6 x 20
##   SO        ATD   CHB     M ‘Alarm Tag Type‘  Flow Level Pressure Temperature
##   <chr>   <dbl> <dbl> <dbl> <chr>            <dbl> <dbl>    <dbl>       <dbl>
## 1 XI-3057     0     0     6 Others               0     0        0           0
## 2 XI-3057     0     0     7 Others               0     0        0           0
## 3 XI-3057     0     0     7 Others               0     0        0           0
## 4 XI-3057     0     0    10 Others               0     0        0           0
## 5 XI-3057     0     0    10 Others               0     0        0           0
## 6 XI-3058     0     0     6 Others               0     0        0           0
## # i 11 more variables: Others <dbl>, H <chr>, ‘Hour:0-6‘ <dbl>,
## #   ‘Hour:7-12‘ <dbl>, ‘Hour:13-18‘ <dbl>, ‘Hour:19-24‘ <dbl>, Week <chr>,
## #   ‘1st Week‘ <dbl>, ‘2nd week‘ <dbl>, ‘3rd week‘ <dbl>, ‘4th week‘ <dbl>
```

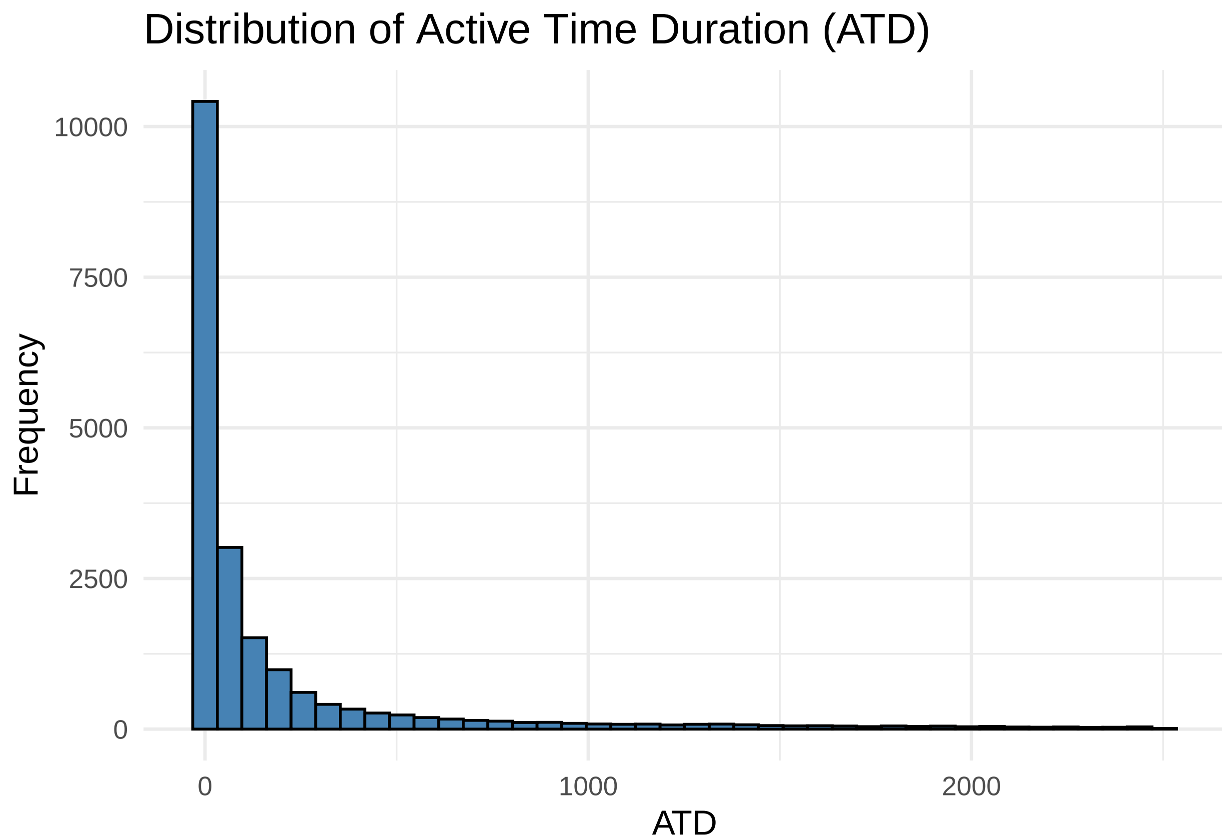
EDA – Alarm Tag Type (Pie Chart)

```
alarm_pie <- data %>%
  count(`Alarm Tag Type`) %>%
  mutate(percent = round(n / sum(n) * 100, 1),
         label = paste0(percent, "%"))

ggplot(alarm_pie, aes(x = "", y = n, fill = `Alarm Tag Type`)) +
  geom_bar(stat = "identity", width = 1) +
  coord_polar(theta = "y") +
  geom_text(aes(label = label), position = position_stack(vjust = 0.5), color = "black", size = 5) +
  labs(title = "Distribution of Alarm Tag Types") +
  theme_void() +
  theme(plot.title = element_text(hjust = 0.5, face = "bold", size = 18))
```



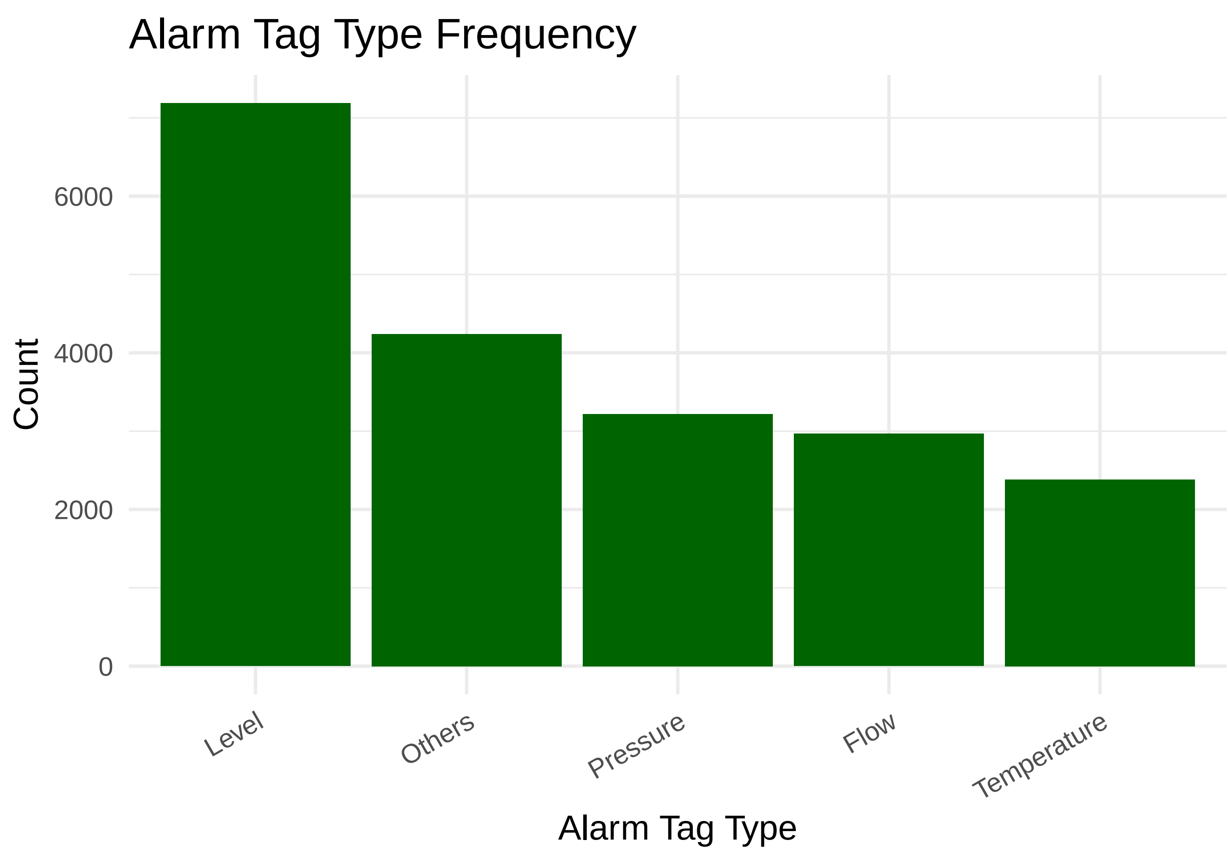
EDA – Distribution of ATD

```
ggplot(data, aes(x = ATD)) +
  geom_histogram(fill = "steelblue", color = "black", bins = 40) +
  labs(title = "Distribution of Active Time Duration (ATD)", x = "ATD", y = "Frequency") +
  theme_minimal(base_size = 13)
```



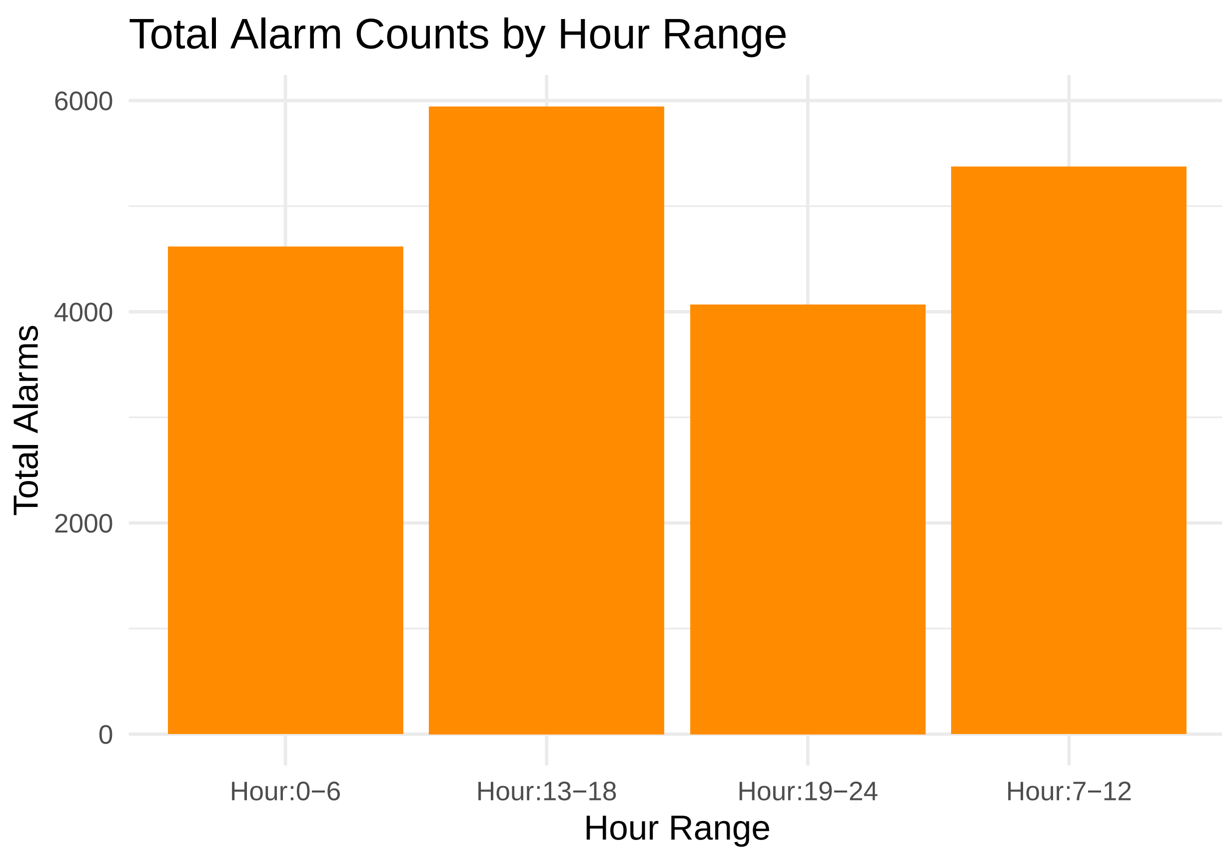
EDA – Alarm Tag Type Frequency

```
alarm_counts <- data %>% count(`Alarm Tag Type`) %>% arrange(desc(n))
ggplot(alarm_counts, aes(x = reorder(`Alarm Tag Type`, -n), y = n)) +
  geom_bar(stat = "identity", fill = "darkgreen") +
  labs(title = "Alarm Tag Type Frequency", x = "Alarm Tag Type", y = "Count") +
  theme_minimal(base_size = 13) +
  theme(axis.text.x = element_text(angle = 30, hjust = 1))
```



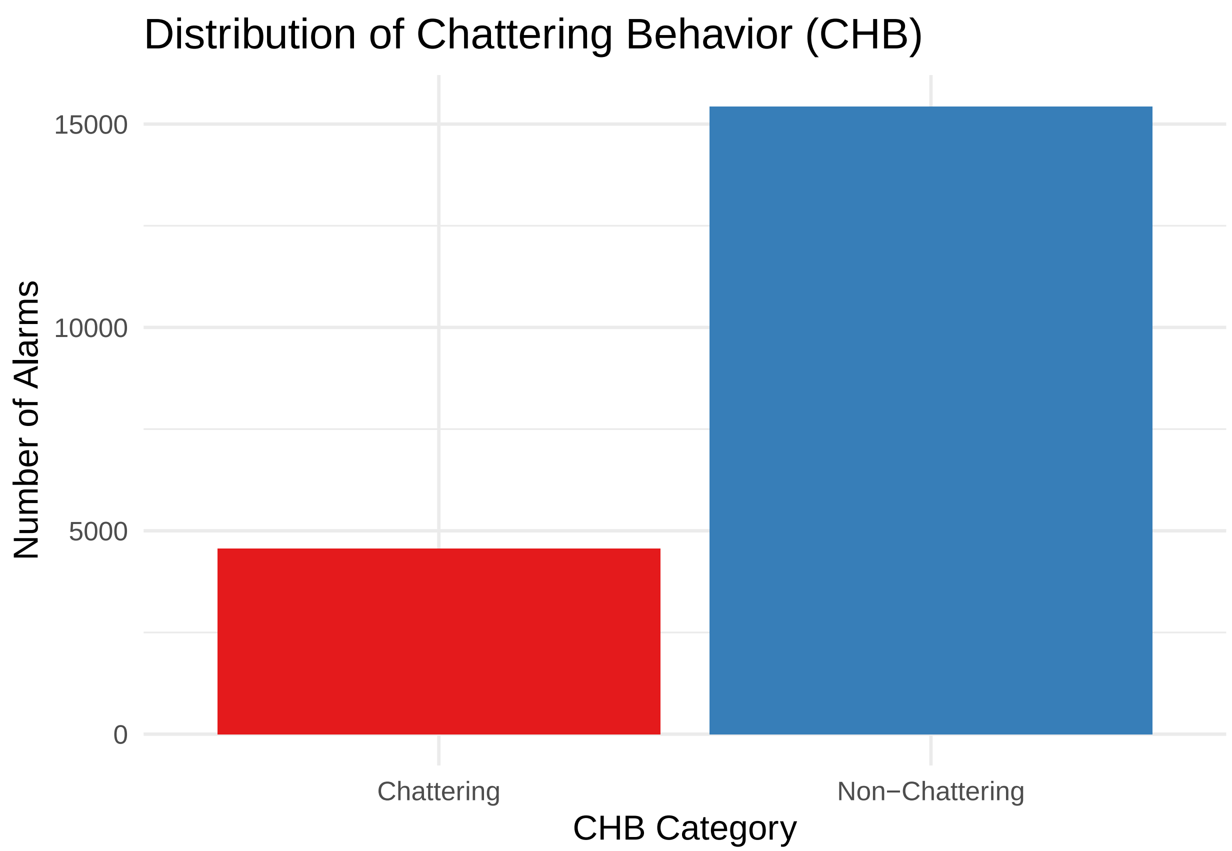
EDA – Hour-Based Alarm Analysis

```
hour_data <- data %>%
  select(`Hour:0-6`, `Hour:7-12`, `Hour:13-18`, `Hour:19-24`) %>%
  pivot_longer(cols = everything(), names_to = "Hour_Range", values_to = "Count") %>%
  group_by(Hour_Range) %>% summarise(Total_Alarms = sum(Count, na.rm = TRUE))

ggplot(hour_data, aes(x = Hour_Range, y = Total_Alarms)) +
  geom_bar(stat = "identity", fill = "#FF8C00") +
  labs(title = "Total Alarm Counts by Hour Range", x = "Hour Range", y = "Total Alarms") +
  theme_minimal(base_size = 13)
```



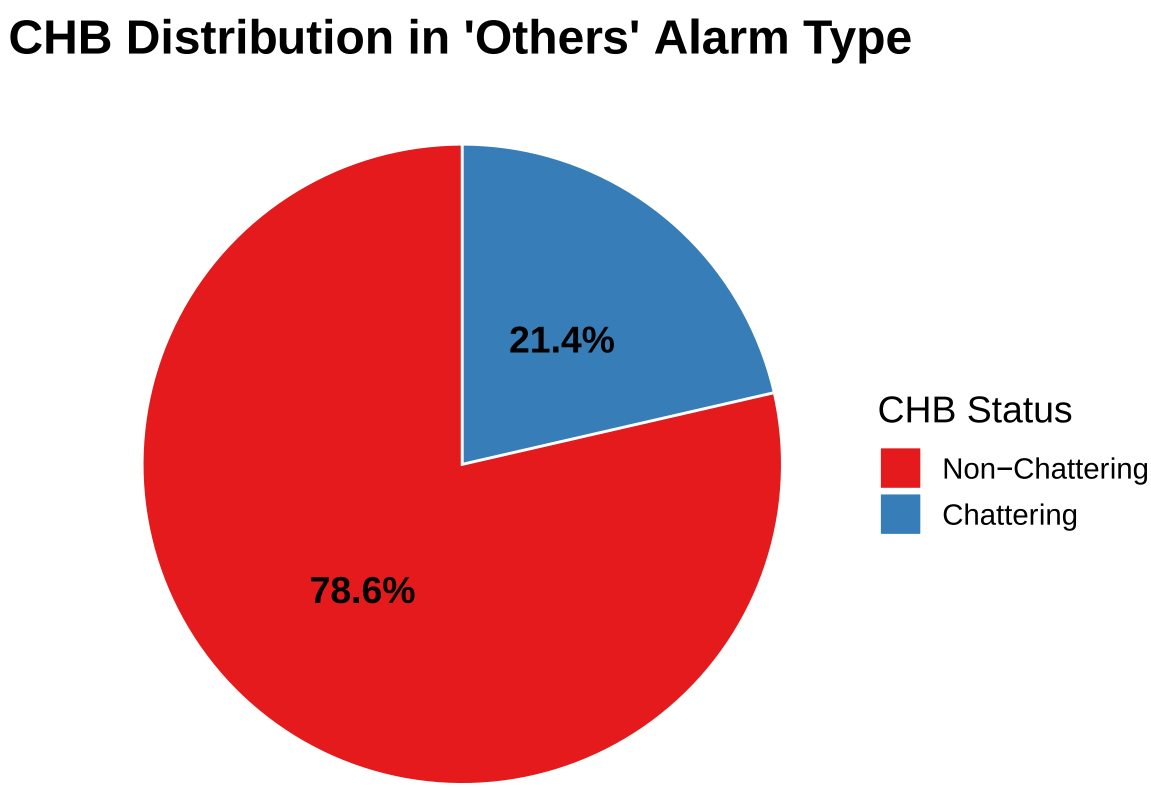
EDA – CHB (Chattering Behavior)

```
chb_counts <- data %>%
  count(CHB) %>%
  mutate(CHB = as.factor(CHB),
         Label = ifelse(CHB == "1", "Chattering", "Non-Chattering"))

ggplot(chb_counts, aes(x = Label, y = n, fill = Label)) +
  geom_bar(stat = "identity") +
  scale_fill_brewer(palette = "Set1") +
  labs(title = "Distribution of Chattering Behavior (CHB)", x = "CHB Category", y = "Number of Alarms")
  theme_minimal(base_size = 13) +
  theme(legend.position = "none")
```



CHB in "Others" Alarm Type – Pie Chart

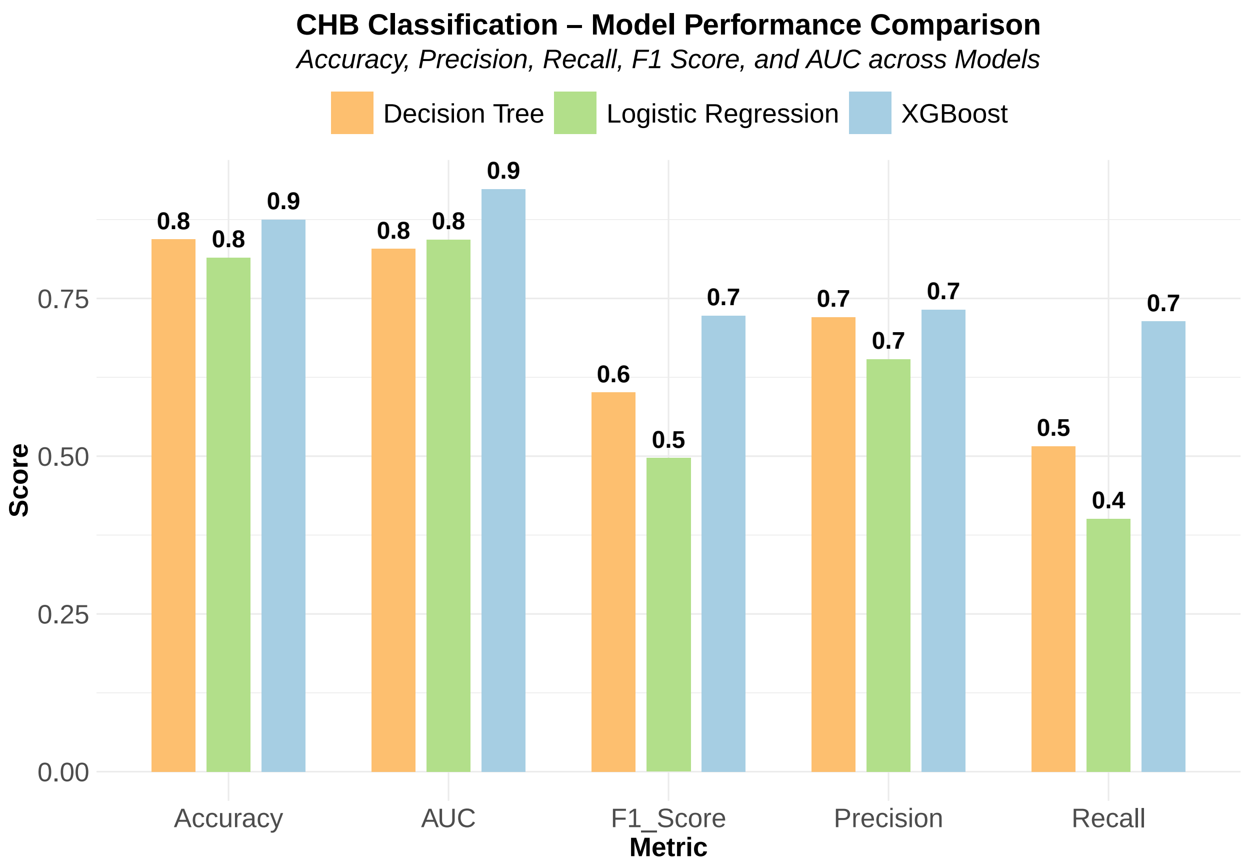
```
others_data <- data %>% filter(`Alarm Tag Type` == "Others")

others_chb_dist <- others_data %>%
  count(CHB) %>%
  mutate(
    CHB = factor(CHB, labels = c("Non-Chattering", "Chattering")),
    percent = round(n / sum(n) * 100, 1),
    label = paste0(percent, "%")
  )

ggplot(others_chb_dist, aes(x = "", y = n, fill = CHB)) +
  geom_bar(stat = "identity", width = 1, color = "white") +
  coord_polar(theta = "y") +
  scale_fill_manual(values = c("Non-Chattering" = "#e41a1c", "Chattering" = "#377eb8")) +
  geom_text(aes(label = label), position = position_stack(vjust = 0.5), size = 5, color = "black", font
  labs(title = "CHB Distribution in 'Others' Alarm Type", fill = "CHB Status") +
  theme_void(base_size = 14) +
  theme(plot.title = element_text(hjust = 0.5, size = 18, face = "bold"))
```



CHB Classification – Model Training and Evaluation

```
data_clean <- data %>%
  select(ATD, CHB, M, `Alarm Tag Type`, H) %>%
  drop_na() %>%
  mutate(CHB = as.factor(CHB), `Alarm Tag Type` = as.factor(`Alarm Tag Type`))

set.seed(42)
train_index <- createDataPartition(data_clean$CHB, p = 0.7, list = FALSE)
train_data <- data_clean[train_index, ]
test_data <- data_clean[-train_index, ]

log_model <- glm(CHB ~ ., data = train_data, family = "binomial")
log_probs <- predict(log_model, newdata = test_data, type = "response")
log_preds <- ifelse(log_probs > 0.5, 1, 0)
log_roc <- roc(as.numeric(as.character(test_data$CHB)), as.numeric(log_probs))

tree_model <- rpart(CHB ~ ., data = train_data, method = "class")
tree_preds <- predict(tree_model, test_data, type = "class")
tree_probs <- predict(tree_model, test_data)[,2]
tree_roc <- roc(as.numeric(as.character(test_data$CHB)), as.numeric(tree_probs))

train_matrix <- model.matrix(CHB ~ . -1, data = train_data)
test_matrix <- model.matrix(CHB ~ . -1, data = test_data)
```

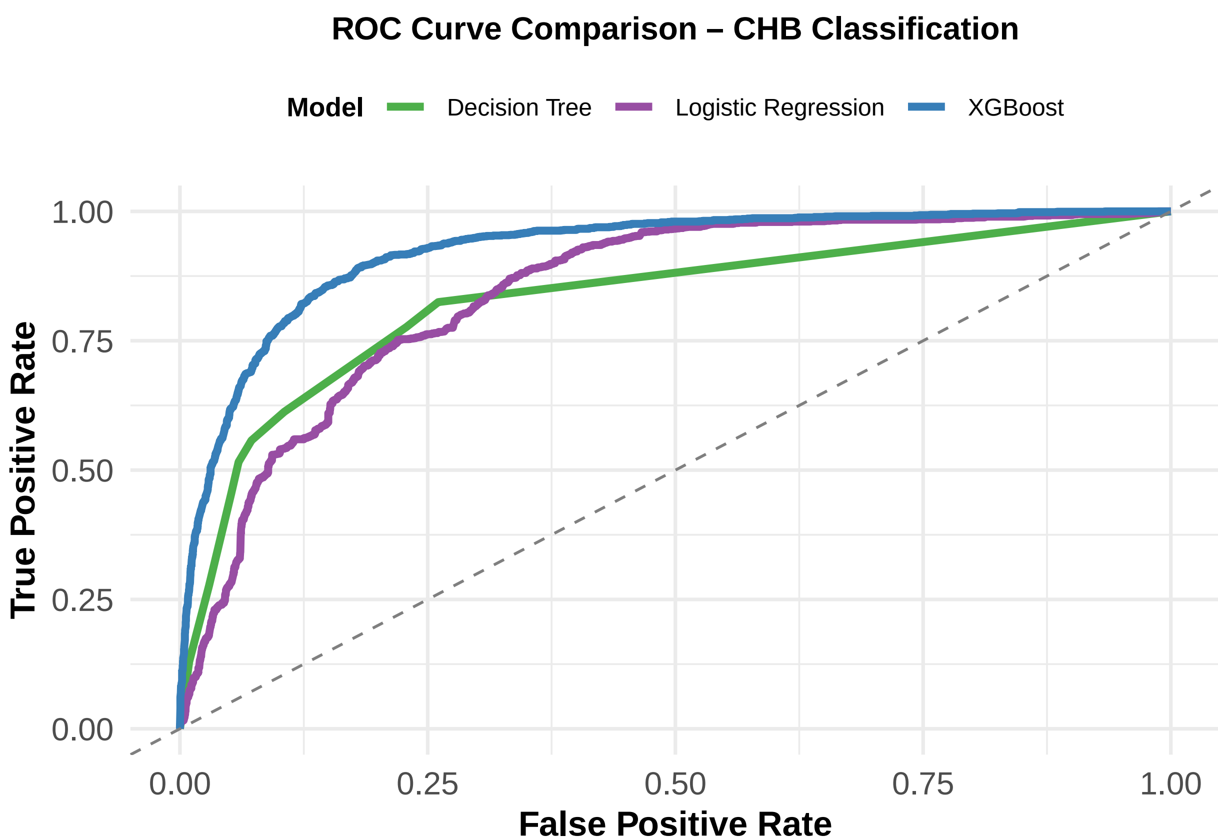
```r
xgb_train <- xgb.DMatrix(data = train_matrix, label = as.numeric(train_data$CHB) - 1)
xgb_test <- xgb.DMatrix(data = test_matrix)
xgb_model <- xgboost(data = xgb_train, nrounds = 50, objective = "binary:logistic", verbose = 0)
xgb_probs <- predict(xgb_model, xgb_test)
xgb_preds <- ifelse(xgb_probs > 0.5, 1, 0)
xgb_roc <- roc(as.numeric(as.character(test_data$CHB)), xgb_probs)

get_metrics <- function(true, pred, probs, roc_obj) {
  cm <- confusionMatrix(as.factor(pred), as.factor(true), positive = "1")
  tibble(
    Accuracy = cm$overall["Accuracy"],
    Precision = cm$byClass["Precision"],
    Recall = cm$byClass["Recall"],
    F1_Score = cm$byClass["F1"],
    AUC = as.numeric(pROC::auc(roc_obj))
  )
}

results <- bind_rows(
  get_metrics(test_data$CHB, log_preds, log_probs, log_roc) %>% mutate(Model = "Logistic Regression"),
  get_metrics(test_data$CHB, tree_preds, tree_probs, tree_roc) %>% mutate(Model = "Decision Tree"),
  get_metrics(test_data$CHB, xgb_preds, xgb_probs, xgb_roc) %>% mutate(Model = "XGBoost")
)

results_long <- results %>%
  pivot_longer(cols = -Model, names_to = "Metric", values_to = "Score")

ggplot(results_long, aes(x = Metric, y = Score, fill = Model)) +
  geom_bar(stat = "identity", position = position_dodge(width = 0.75), width = 0.6) +
  geom_text(aes(label = round(Score, 1)), position = position_dodge(width = 0.75), vjust = -0.6, size =
  scale_fill_manual(values = c("Logistic Regression" = "#b2df8a", "Decision Tree" = "#fdbf6f", "XGBoost
  labs(title = "CHB Classification - Model Performance Comparison", subtitle = "Accuracy, Precision, Re
  theme_minimal(base_size = 6) +
  theme(plot.title = element_text(hjust = 0.5, face = "bold", size = 11),
        plot.subtitle = element_text(hjust = 0.5, size = 10, face = "italic"),
        axis.title = element_text(size = 10, face = "bold"),
        axis.text = element_text(size = 10),
        legend.position = "top",
        legend.title = element_blank(),
        legend.text = element_text(size = 10))
```



ROC Curve – CHB Classification

```
log_df <- data.frame(FPR = rev(1 - log_roc$specificities), TPR = rev(log_roc$sensitivities), Model = "Lo
tree_df <- data.frame(FPR = rev(1 - tree_roc$specificities), TPR = rev(tree_roc$sensitivities), Model =
xgb_df <- data.frame(FPR = rev(1 - xgb_roc$specificities), TPR = rev(xgb_roc$sensitivities), Model = "XG

roc_combined <- bind_rows(log_df, tree_df, xgb_df)

ggplot(roc_combined, aes(x = FPR, y = TPR, color = Model)) +
  geom_line(size = 1.4) +
  geom_abline(linetype = "dashed", color = "gray50") +
  scale_color_manual(values = c("XGBoost" = "#377eb8", "Decision Tree" = "#4daf4a", "Logistic Regressio
  labs(
    title = "ROC Curve Comparison - CHB Classification",
    x = "False Positive Rate",
    y = "True Positive Rate",
    color = "Model"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold", size = 12),
    axis.title = element_text(size = 13, face = "bold"),
    axis.text = element_text(size = 12),
    legend.title = element_text(size = 10, face = "bold"),
```

```
    legend.text = element_text(size = 9),
    legend.position = "top"
  )
```



ATD Prediction and Residual Analysis

This section compares linear regression and XGBoost regressor models in predicting the Active Time Duration (ATD). It also includes residual analysis to evaluate prediction accuracy across the range of ATD values.

```
# Load additional library
library(Metrics)

# Prepare the data
data_clean_reg <- data %>%
  select(ATD, CHB, M, `Alarm Tag Type`, H) %>%
  drop_na() %>%
  mutate(`Alarm Tag Type` = as.factor(`Alarm Tag Type`))

# Split into training and test sets
set.seed(123)
train_index <- createDataPartition(data_clean_reg$ATD, p = 0.7, list = FALSE)
train_data <- data_clean_reg[train_index, ]
test_data <- data_clean_reg[-train_index, ]
```

```r
# Train Linear Regression
linear_model <- lm(ATD ~ ., data = train_data)
linear_preds <- predict(linear_model, newdata = test_data)

# Train XGBoost Regressor
train_matrix_reg <- model.matrix(ATD ~ . -1, data = train_data)
test_matrix_reg <- model.matrix(ATD ~ . -1, data = test_data)
xgb_train_reg <- xgb.DMatrix(data = train_matrix_reg, label = train_data$ATD)
xgb_test_reg <- xgb.DMatrix(data = test_matrix_reg)
xgb_model_reg <- xgboost(data = xgb_train_reg, nrounds = 50, objective = "reg:squarederror", verbose = (
xgb_preds <- predict(xgb_model_reg, xgb_test_reg)

# Evaluation
rmse_linear <- rmse(test_data$ATD, linear_preds)
rmse_xgb <- rmse(test_data$ATD, xgb_preds)
mae_linear <- mae(test_data$ATD, linear_preds)
mae_xgb <- mae(test_data$ATD, xgb_preds)

cat("Linear Regression - RMSE:", round(rmse_linear, 2), " MAE:", round(mae_linear, 2), "\n")
```

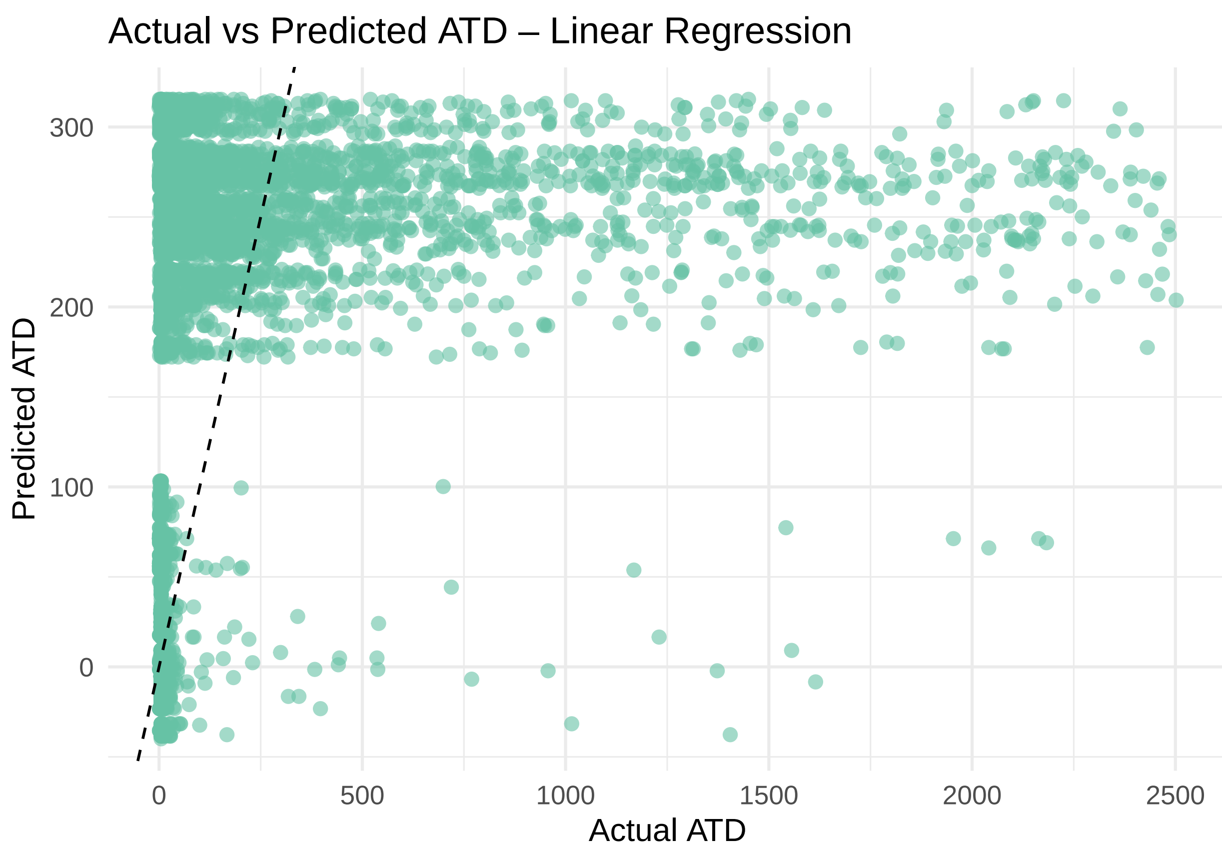
```
## Linear Regression - RMSE: 407.53  MAE: 240.02
```

```r
cat("XGBoost Regressor - RMSE:", round(rmse_xgb, 2), " MAE:", round(mae_xgb, 2), "\n")
```

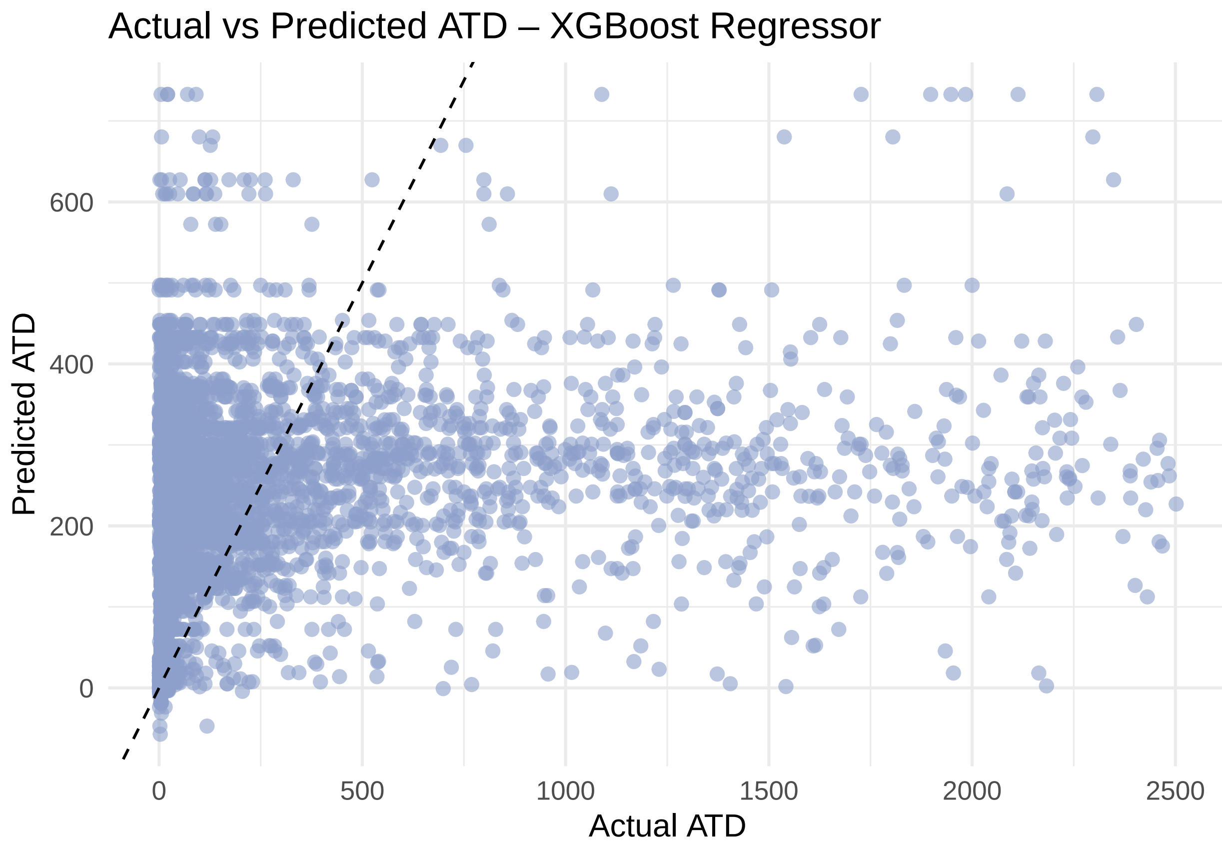
```
## XGBoost Regressor - RMSE: 404.03  MAE: 231.82
```

Actual vs Predicted Plots

```r
# Combine predictions
results_plot <- data.frame(
  Actual = test_data$ATD,
  Linear_Regression = linear_preds,
  XGBoost = xgb_preds
)

# Linear Regression Plot
ggplot(results_plot, aes(x = Actual, y = Linear_Regression)) +
  geom_point(color = "#66c2a5", alpha = 0.6, size = 2) +
  geom_abline(slope = 1, intercept = 0, linetype = "dashed", color = "black") +
  labs(title = "Actual vs Predicted ATD - Linear Regression", x = "Actual ATD", y = "Predicted ATD") +
  theme_minimal(base_size = 12)
```

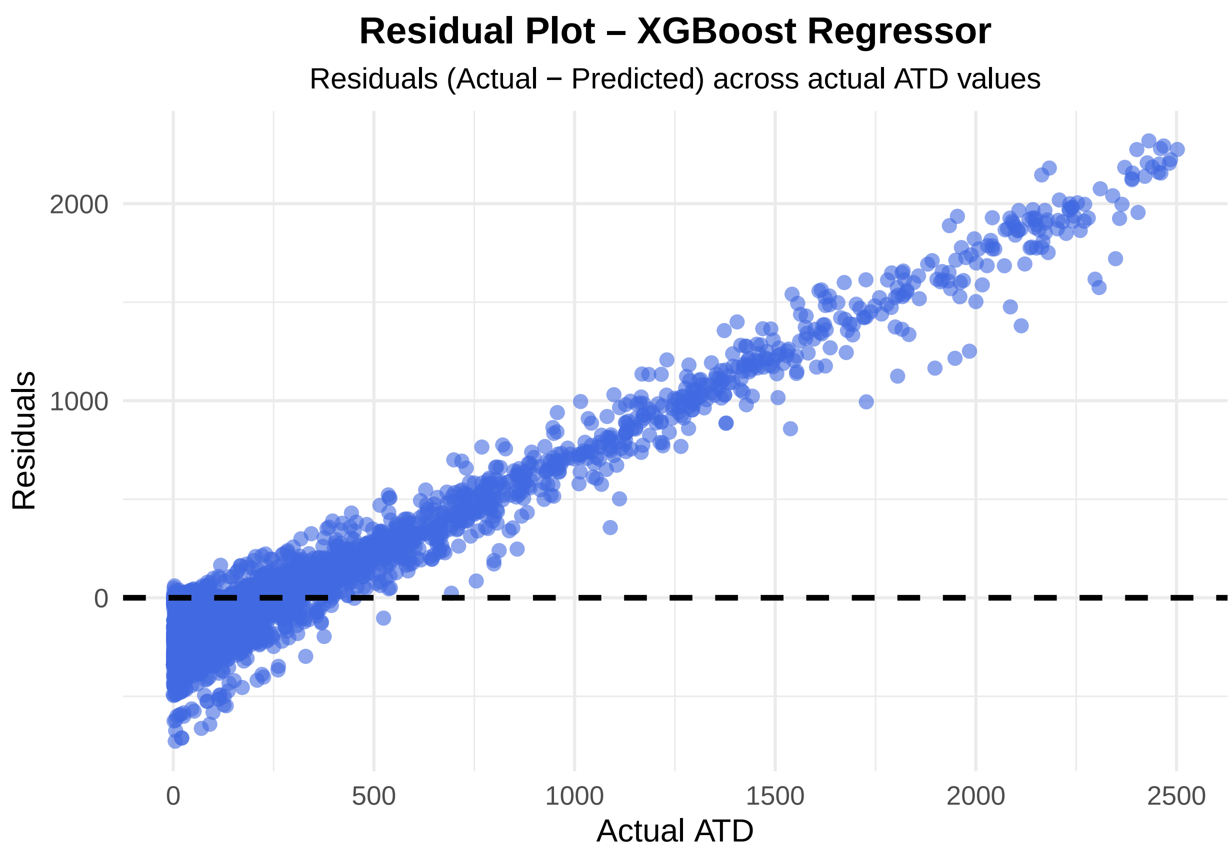



```r
# XGBoost Regressor Plot
ggplot(results_plot, aes(x = Actual, y = XGBoost)) +
  geom_point(color = "#8da0cb", alpha = 0.6, size = 2) +
  geom_abline(slope = 1, intercept = 0, linetype = "dashed", color = "black") +
  labs(title = "Actual vs Predicted ATD - XGBoost Regressor", x = "Actual ATD", y = "Predicted ATD") +
  theme_minimal(base_size = 12)
```



Residual Plot

```r
# Residual analysis for XGBoost
residuals <- test_data$ATD - xgb_preds
ggplot(data.frame(Actual = test_data$ATD, Residual = residuals),
       aes(x = Actual, y = Residual)) +
  geom_point(color = "royalblue", alpha = 0.6, size = 2) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "black", linewidth = 1) +
  labs(
    title = "Residual Plot - XGBoost Regressor",
    subtitle = "Residuals (Actual - Predicted) across actual ATD values",
    x = "Actual ATD",
    y = "Residuals"
  ) +
  theme_minimal(base_size = 12) +
  theme(
    plot.title = element_text(size = 14, face = "bold", hjust = 0.5),
    plot.subtitle = element_text(size = 11, hjust = 0.5),
    axis.title = element_text(size = 12),
    axis.text = element_text(size = 10)
  )
```



Residual Plot – XGBoost Regressor
Residuals (Actual − Predicted) across actual ATD values
Residuals
2000
1000
0
0
500
1000
1500
2000
2500
Actual ATD