House Prediction

Team

2024-12-01

R Markdown

```
# Load libraries
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(caret)
```

```
##      lattice
##
##    'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(gbm)
```

```
## Loaded gbm 2.2.2
## This version of gbm is no longer under development. Consider transitioning to gbm3, https://github.co
```

```
library(xgboost)
```

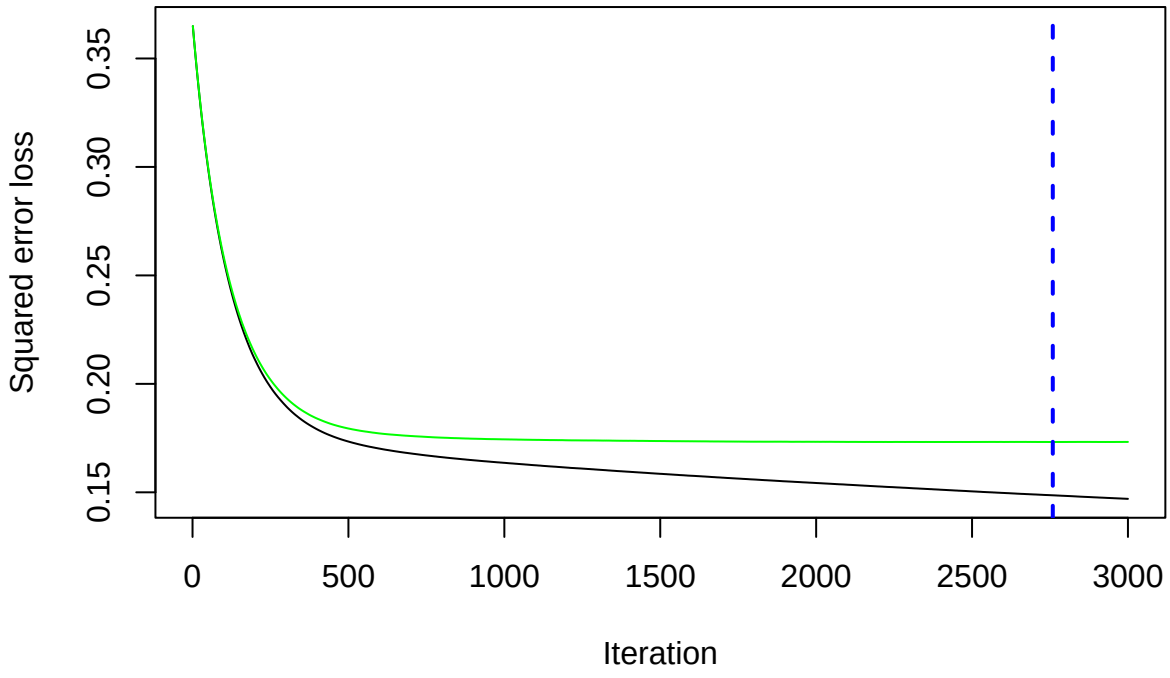
```
##
##    'xgboost'
##
## The following object is masked from 'package:dplyr':
##
##     slice
```

```r
library(randomForest)
```

```
## randomForest 4.7-1.2
## Type rfNews() to see new features/changes/bug fixes.
##
##     'randomForest'
##
## The following object is masked from 'package:dplyr':
##
##     combine
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
library(ggplot2)

# Load data
housing_data <- read.csv("C:/Users/Bowen/Downloads/437/CPTS_437_data/whitman_property_details.csv")

# Data cleaning and feature engineering
clean_data <- housing_data %>%
  mutate(
    Total_Area = as.numeric(gsub(",", "", ifelse(Total_Area == "None", NA, Total_Area))),
    Year_Built = as.numeric(ifelse(Year_Built == "None", NA, Year_Built)),
    Total_Value = as.numeric(gsub(",", "", ifelse(Total_Value == "None", NA, Total_Value))),
    Bedrooms = as.numeric(ifelse(Bedrooms == "None", NA, Bedrooms)),
    Bathrooms = as.numeric(ifelse(Bathrooms == "None", NA, Bathrooms)),
    Garage_Stalls = as.numeric(ifelse(Garage_Stalls %in% c("None", "Block"), 0, Garage_Stalls))
  ) %>%
  filter(!is.na(Total_Value) & !is.na(Total_Area) & !is.na(Year_Built)) %>%
  mutate(
    log_value = log(Total_Value + 1),
    log_area = log(Total_Area + 1),
    age = 2024 - Year_Built,
    has_garage = ifelse(is.na(Garage_Stalls), 0, 1),
    rooms_per_area = (Bedrooms + Bathrooms) / log_area,
    condition_score = case_when(
      grepl("3.0", Condition) ~ 3.0,
      grepl("3.5", Condition) ~ 3.5,
      grepl("4.0", Condition) ~ 4.0,
      TRUE ~ 3.0
    ),
    age_condition_interaction = age * condition_score
  ) %>%
  filter(
    Total_Value > quantile(Total_Value, 0.03) & Total_Value < quantile(Total_Value, 0.97),
    Total_Area > quantile(Total_Area, 0.03) & Total_Area < quantile(Total_Area, 0.97)
  ) %>%
  select(log_value, log_area, age, has_garage, rooms_per_area, Bathrooms,
         condition_score, age_condition_interaction) %>%
  na.omit()
```

```r
# Split data
set.seed(100)
train_index <- createDataPartition(clean_data$log_value, p = 0.8, list = FALSE)
train_data <- clean_data[train_index, ]
test_data <- clean_data[-train_index, ]

# GBM Model
gbm_model <- gbm(
  log_value ~ .,
  data = train_data,
  distribution = "gaussian",
  n.trees = 3000,
  interaction.depth = 8,
  shrinkage = 0.005,
  n.minobsinnode = 8,
  bag.fraction = 0.8,
  cv.folds = 5
)
best_iter_gbm <- gbm.perf(gbm_model, method = "cv")
```




```r
# XGBoost Model
train_matrix <- xgb.DMatrix(data = as.matrix(train_data %>% select(-log_value)), label = train_data$log
test_matrix <- xgb.DMatrix(data = as.matrix(test_data %>% select(-log_value)))
xgb_model <- xgboost(
  data = train_matrix,
```

```
  objective = "reg:squarederror",
  nrounds = 2000,
  max_depth = 6,
  eta = 0.01,
  subsample = 0.8,
  colsample_bytree = 0.8,
  verbose = 0
)

# Random Forest Model
rf_model <- randomForest(
  log_value ~ .,
  data = train_data,
  ntree = 500,
  mtry = floor(sqrt(ncol(train_data))),
  importance = TRUE
)

# Predictions and Metrics
gbm_predictions <- exp(predict(gbm_model, test_data, n.trees = best_iter_gbm)) - 1
xgb_predictions <- exp(predict(xgb_model, test_matrix)) - 1
rf_predictions <- exp(predict(rf_model, test_data)) - 1
actual_values <- exp(test_data$log_value) - 1

metrics <- function(predictions, actual) {
  rmse <- sqrt(mean((predictions - actual)^2))
  r2 <- 1 - sum((actual - predictions)^2) / sum((actual - mean(actual))^2)
  mae <- mean(abs(predictions - actual))
  return(list(RMSE = rmse, R2 = r2, MAE = mae))
}

gbm_metrics <- metrics(gbm_predictions, actual_values)
xgb_metrics <- metrics(xgb_predictions, actual_values)
rf_metrics <- metrics(rf_predictions, actual_values)

# Print metrics
cat("\nGBM Metrics:\n")
```

```
##
## GBM Metrics:
```

```
print(gbm_metrics)
```

```
## $RMSE
## [1] 86473.55
##
## $R2
## [1] 0.5852003
##
## $MAE
## [1] 66503.06
```

```r
cat("\nXGBoost Metrics:\n")
```

```
##
## XGBoost Metrics:
```

```r
print(xgb_metrics)
```

```
## $RMSE
## [1] 87223.42
##
## $R2
## [1] 0.5779751
##
## $MAE
## [1] 66200.3
```

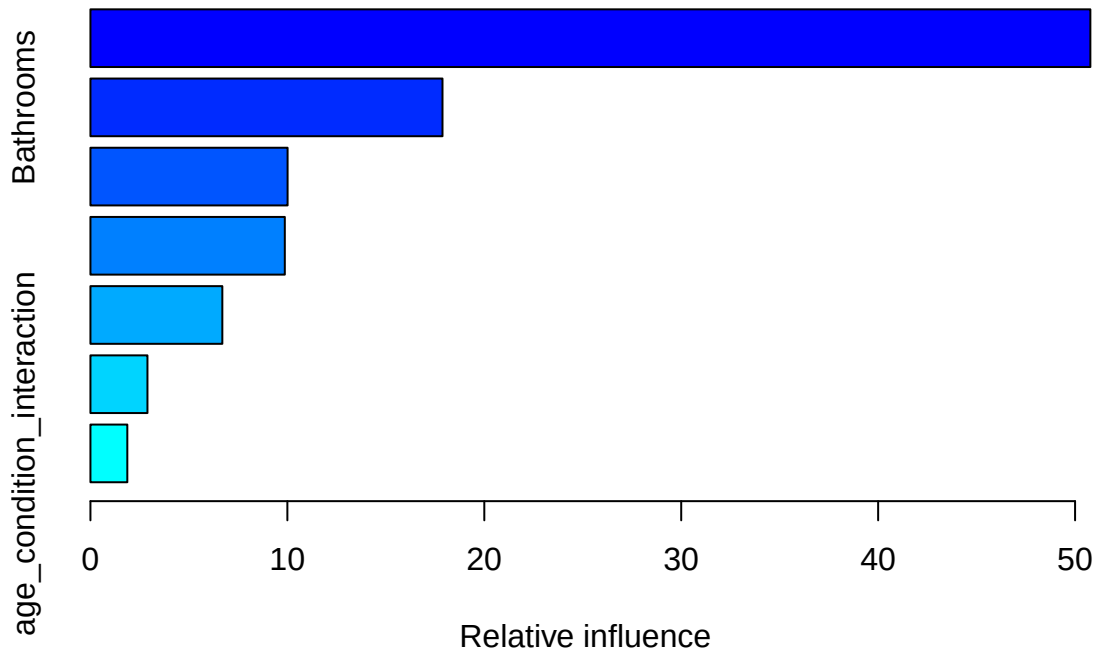
```r
cat("\nRandom Forest Metrics:\n")
```

```
##
## Random Forest Metrics:
```

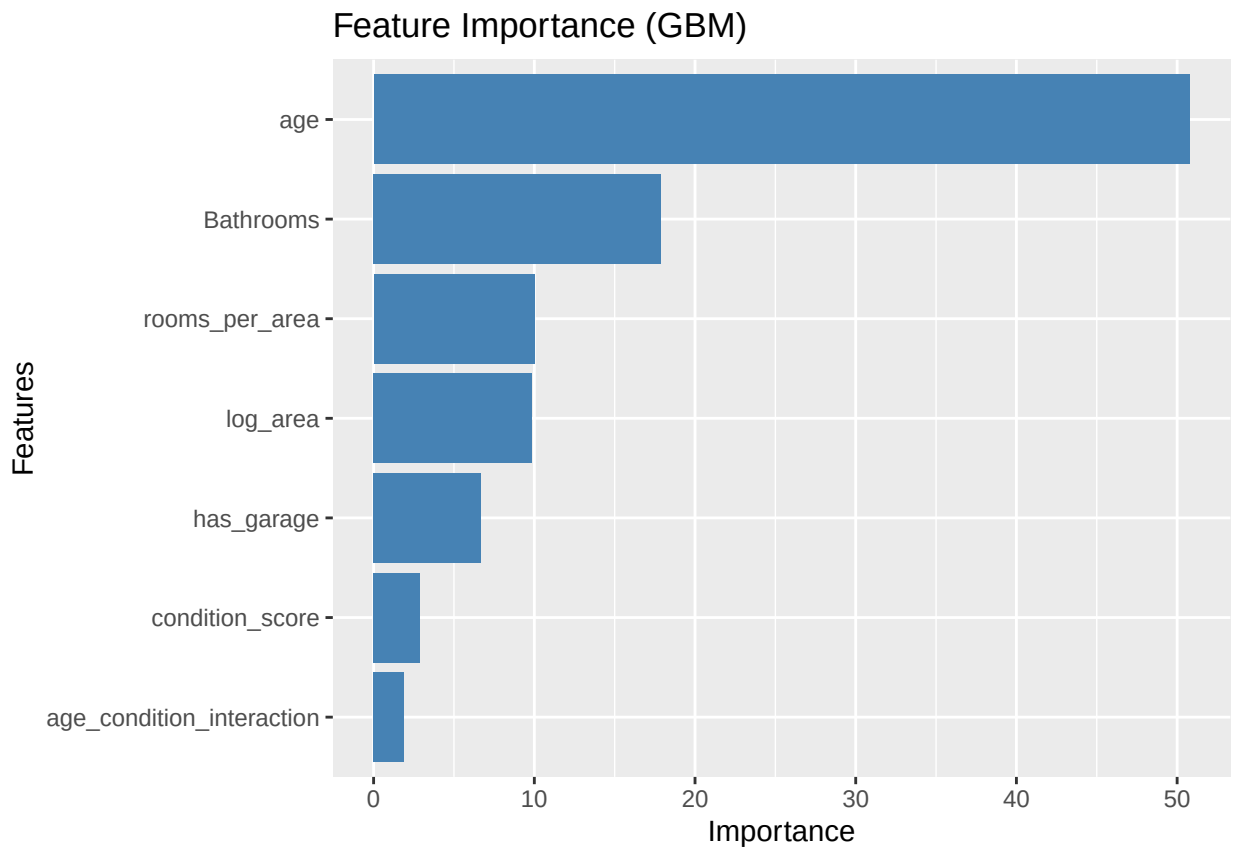
```r
print(rf_metrics)
```

```
## $RMSE
## [1] 87101.42
##
## $R2
## [1] 0.5791548
##
## $MAE
## [1] 66275.61
```

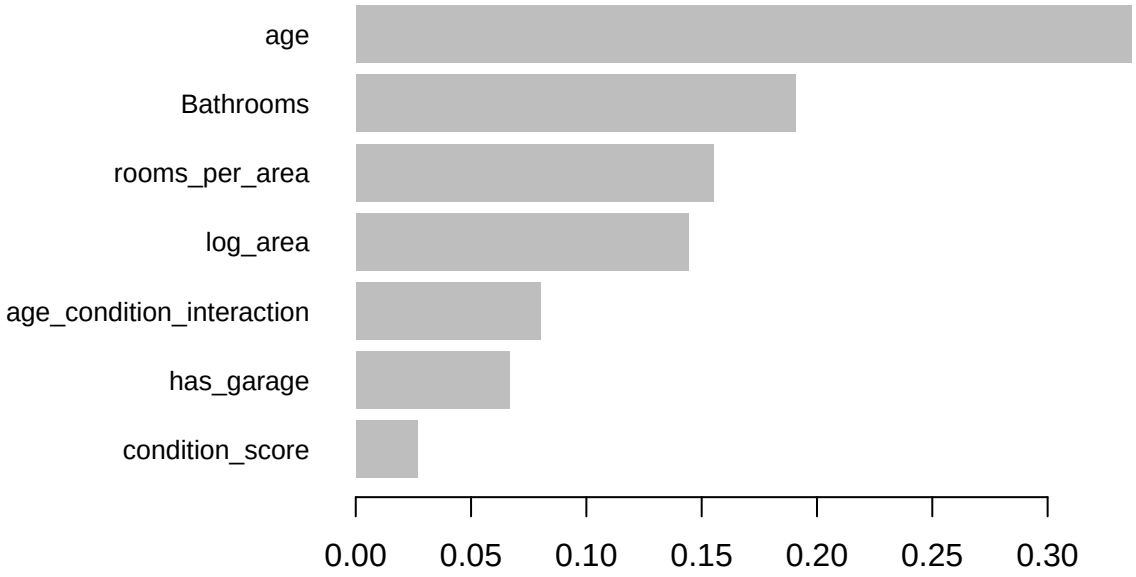
```r
# Visualization
importance_gbm <- summary(gbm_model, n.trees = best_iter_gbm)
```



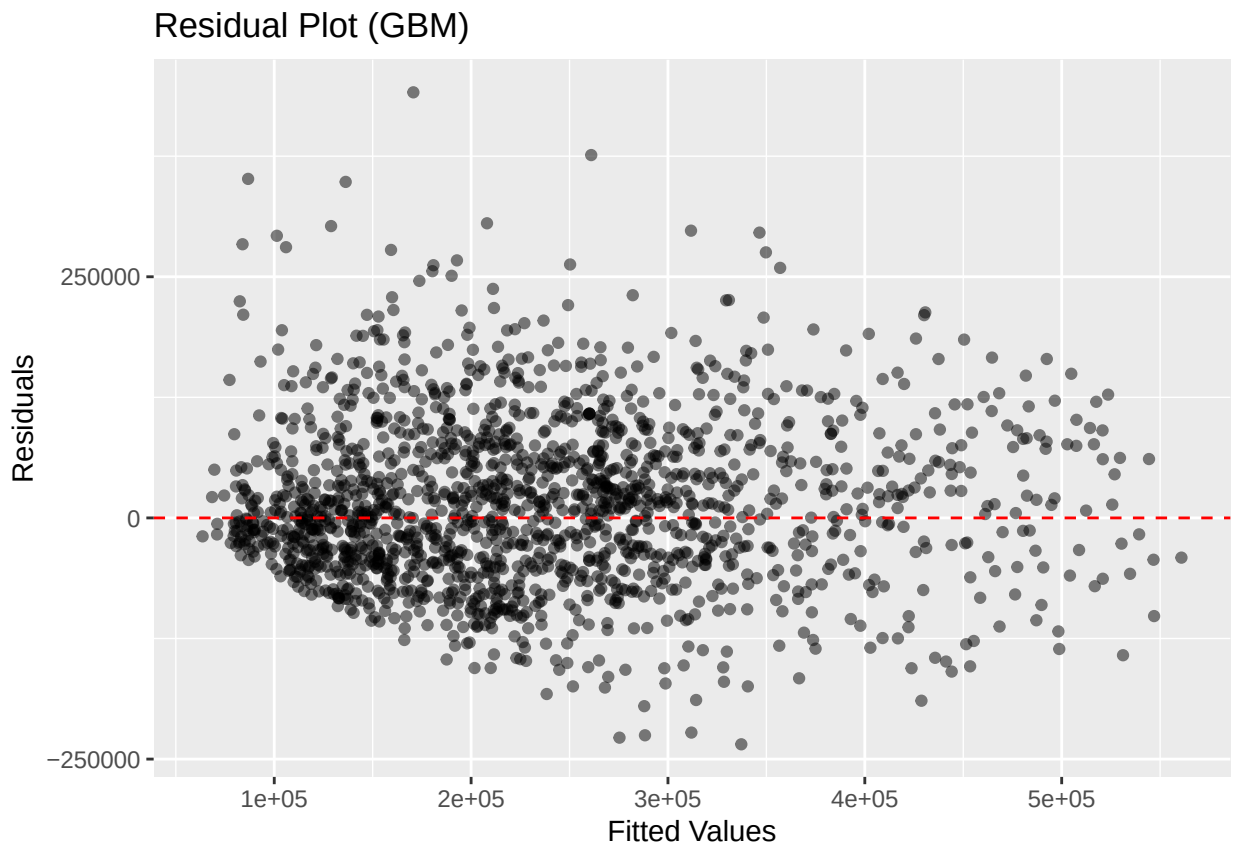

```r
importance_xgb <- xgb.importance(model = xgb_model)

# Feature importance plots
ggplot(data.frame(Feature = importance_gbm$var, Importance = importance_gbm$rel.inf), aes(x = reorder(F
  geom_bar(stat = "identity", fill = "steelblue") +
  coord_flip() +
  labs(title = "Feature Importance (GBM)", x = "Features", y = "Importance")
```



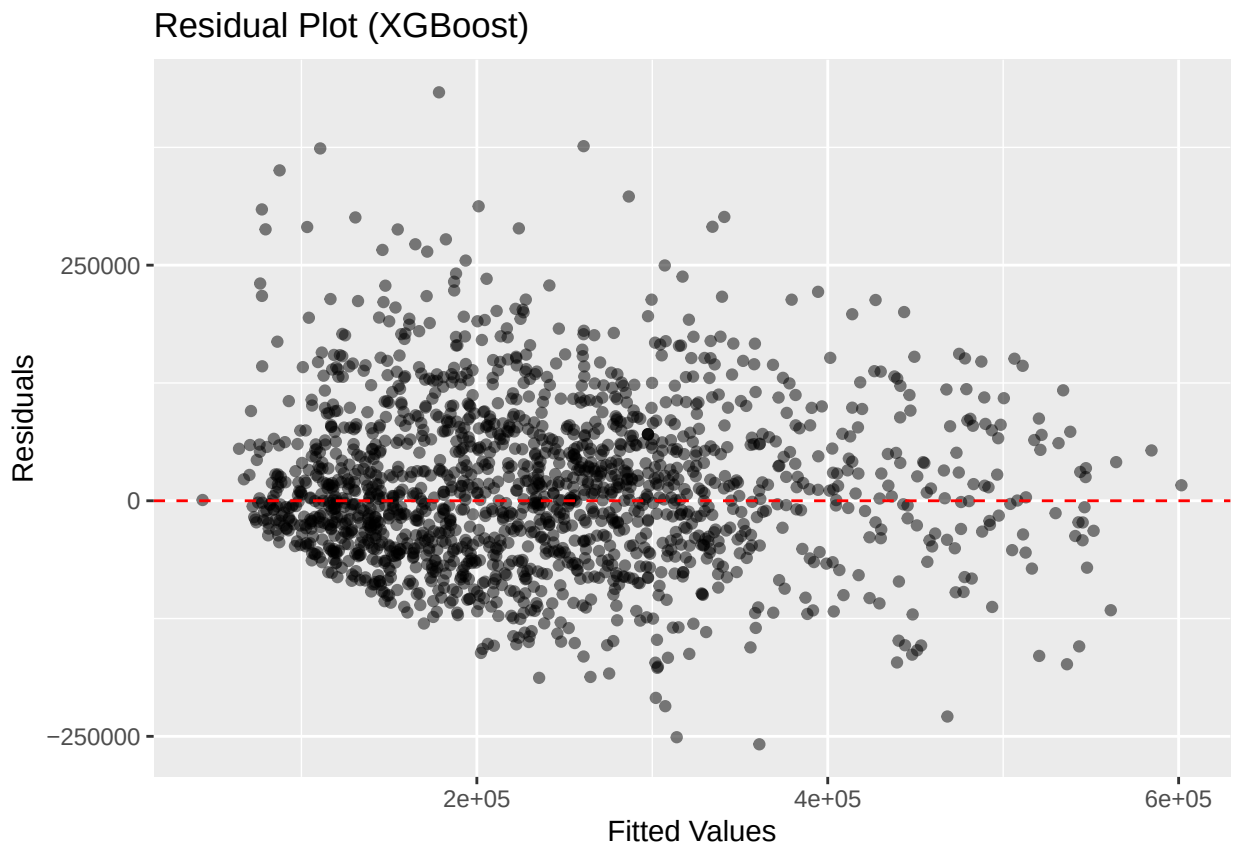

```
xgb.plot.importance(importance_xgb)
```



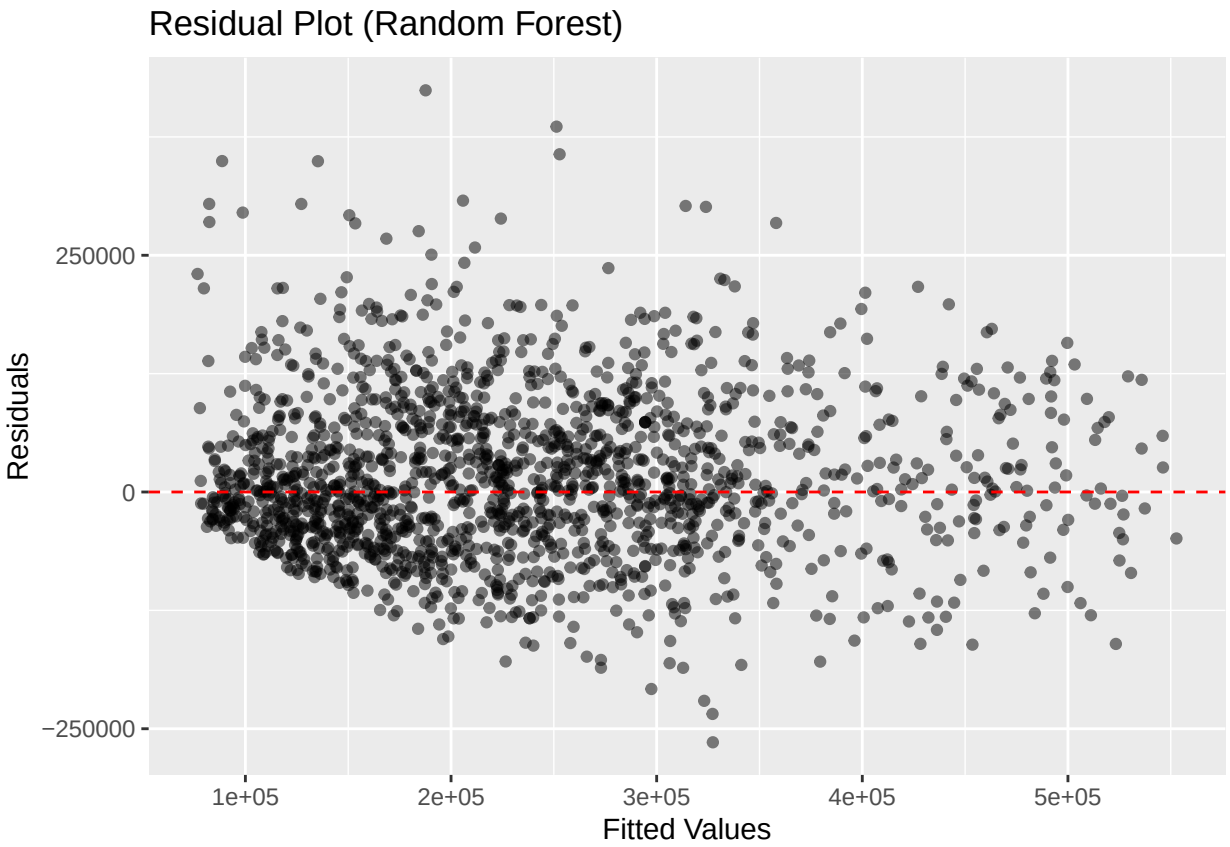

```r
# Residual plots
gbm_residuals <- actual_values - gbm_predictions
xgb_residuals <- actual_values - xgb_predictions
rf_residuals <- actual_values - rf_predictions

# GBM Residual Plot
ggplot(data.frame(fitted = gbm_predictions, residuals = gbm_residuals), aes(x = fitted, y = residuals)) +
  geom_point(alpha = 0.5) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  labs(title = "Residual Plot (GBM)", x = "Fitted Values", y = "Residuals")
```




```r
# XGBoost Residual Plot
ggplot(data.frame(fitted = xgb_predictions, residuals = xgb_residuals), aes(x = fitted, y = residuals)) +
  geom_point(alpha = 0.5) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  labs(title = "Residual Plot (XGBoost)", x = "Fitted Values", y = "Residuals")
```




```r
# Random Forest Residual Plot
ggplot(data.frame(fitted = rf_predictions, residuals = rf_residuals), aes(x = fitted, y = residuals)) +
  geom_point(alpha = 0.5) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  labs(title = "Residual Plot (Random Forest)", x = "Fitted Values", y = "Residuals")
```


Residual Plot (Random Forest)
Residuals
250000
0
−250000
1e+05
2e+05
3e+05
4e+05
5e+05
Fitted Values