House Prediction

Team

2024-12-01

R Markdown

```r
# 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 error
```

```r
library(caret)
```

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

```r
library(gbm)
```

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

```r
# Read and prepare data
housing_data <- read.csv("C:/Users/Bowen/Downloads/437/CPTS_437_data/whitman_property_details.csv")

# Enhanced data cleaning with more features
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))
```

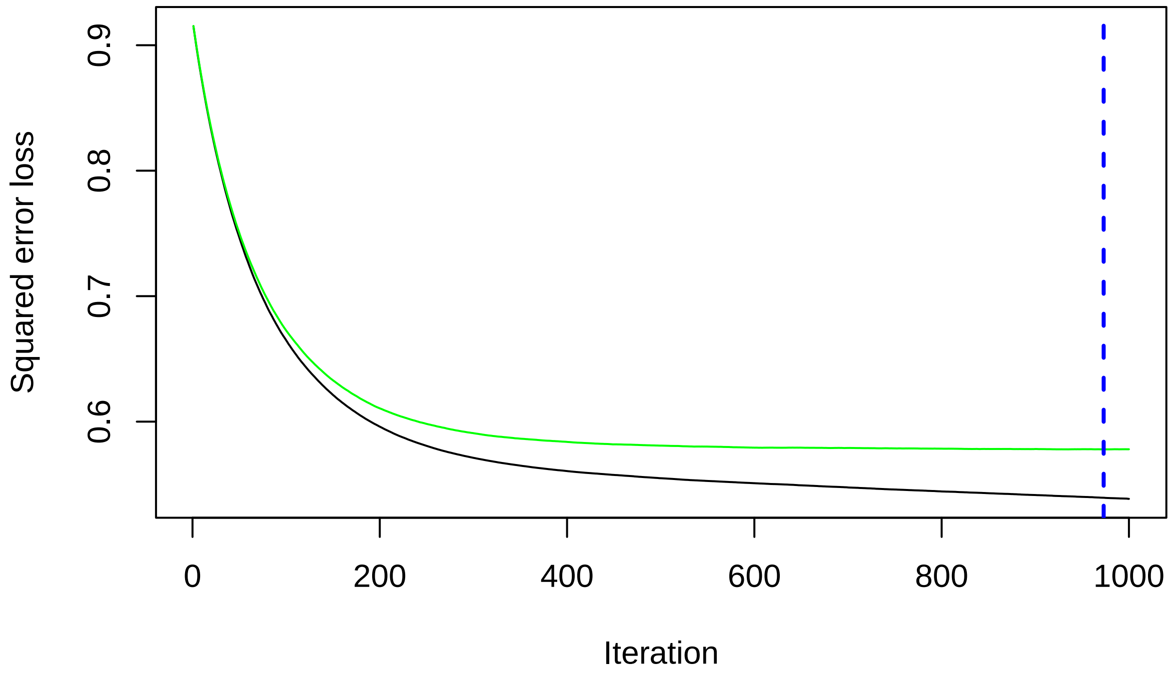
```
## Warning: There were 4 warnings in `mutate()`.
## The first warning was:
## i In argument: `Total_Area = as.numeric(gsub(",", "", ifelse(Total_Area ==
##   "None", NA, Total_Area)))`.
## Caused by warning:
## !      NA
## i Run `dplyr::last_dplyr_warnings()` to see the 3 remaining warnings.
```

```r
# Enhanced feature engineering with additional features
model_data <- clean_data %>%
  mutate(
    log_value = log(Total_Value + 1),
    log_area = log(Total_Area + 1),
    age = 2024 - Year_Built,
    age_squared = age^2,
    has_garage = ifelse(is.na(Garage_Stalls), 0, 1),
    garage_value = ifelse(is.na(Garage_Stalls), 0, Garage_Stalls),
    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
    ),
    bedrooms = ifelse(is.na(Bedrooms), median(Bedrooms, na.rm=TRUE), Bedrooms),
    bathrooms = ifelse(is.na(Bathrooms), median(Bathrooms, na.rm=TRUE), Bathrooms),
    rooms_per_area = (bedrooms + bathrooms) / log_area,
    age_condition_interaction = age * condition_score
  ) %>%
  select(log_value, log_area, age, age_squared, has_garage, garage_value,
         bedrooms, bathrooms, condition_score, rooms_per_area,
         age_condition_interaction) %>%
  na.omit()

# Split data with stratification
set.seed(123)
train_index <- createDataPartition(model_data$log_value, p = 0.8, list = FALSE)
train_data <- model_data[train_index, ]
test_data <- model_data[-train_index, ]

# Train GBM model with tuned parameters
gbm_model <- gbm(
  log_value ~ .,
  data = train_data,
  distribution = "gaussian",
  n.trees = 1000,
  interaction.depth = 4,
  shrinkage = 0.01,
  n.minobsinnode = 10,
  bag.fraction = 0.8,  # Added bagging
  cv.folds = 5         # Added cross-validation
)

# Find optimal number of trees
best_iter <- gbm.perf(gbm_model, method = "cv")
```



```
print(paste("Optimal number of trees:", best_iter))
```

```
## [1] "Optimal number of trees: 973"
```

```
# Make predictions using optimal number of trees
predictions <- exp(predict(gbm_model, test_data, n.trees = best_iter)) - 1
actual_values <- exp(test_data$log_value) - 1

# Calculate metrics
rmse <- sqrt(mean((predictions - actual_values)^2))
r2 <- 1 - sum((actual_values - predictions)^2) /
         sum((actual_values - mean(actual_values))^2)
mae <- mean(abs(predictions - actual_values))
```

```r
# Print comprehensive metrics
print(paste("RMSE:", format(rmse, scientific = FALSE)))
```

```
## [1] "RMSE: 372187.9"
```

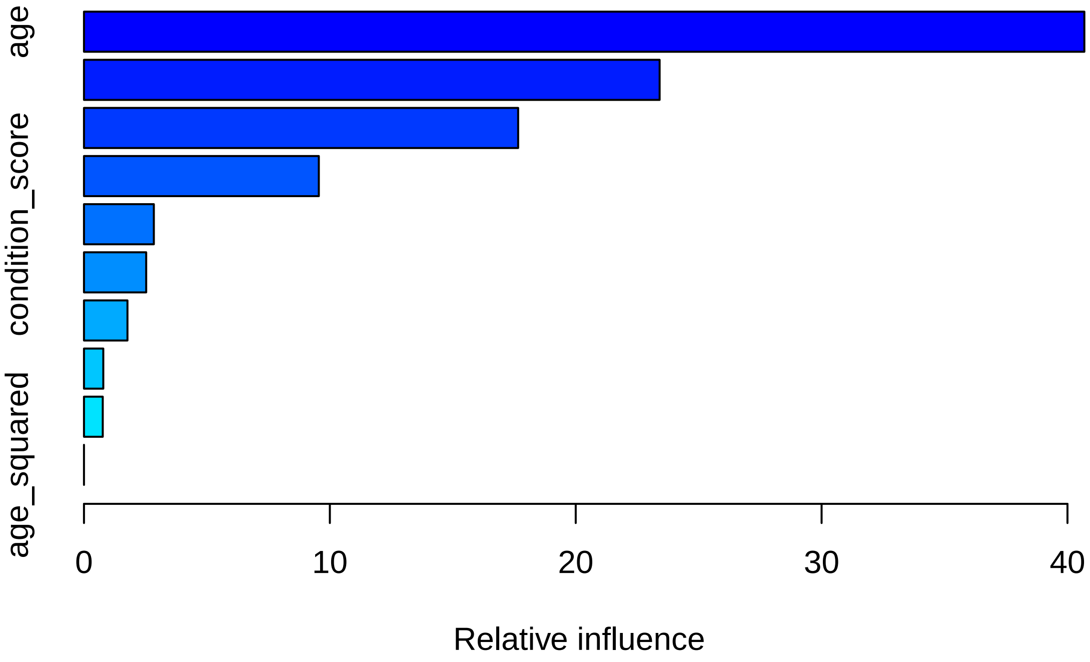
```r
print(paste("R-squared:", round(r2, 3)))
```

```
## [1] "R-squared: 0.631"
```

```r
print(paste("MAE:", format(mae, scientific = FALSE)))
```

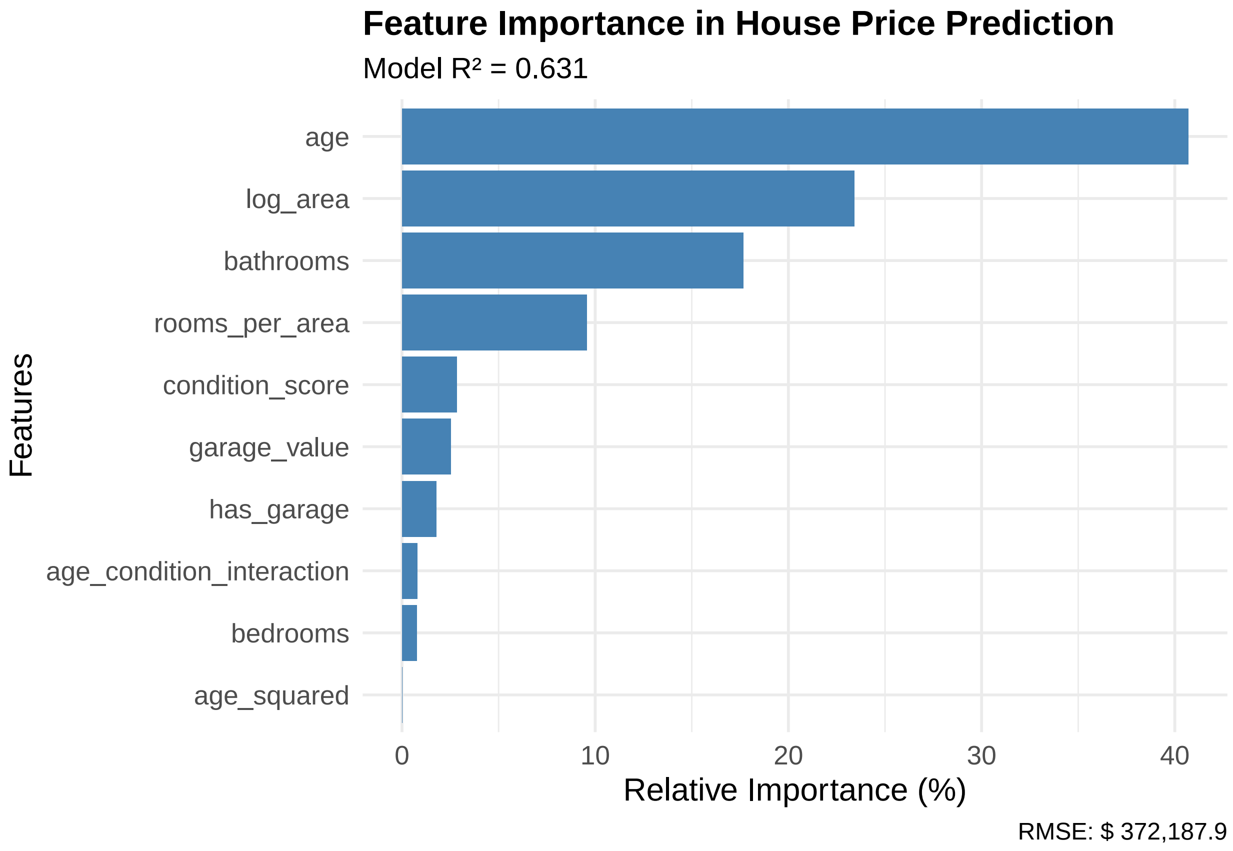
```
## [1] "MAE: 107127.8"
```

```r
# Enhanced visualization
importance <- summary(gbm_model, n.trees = best_iter)
```

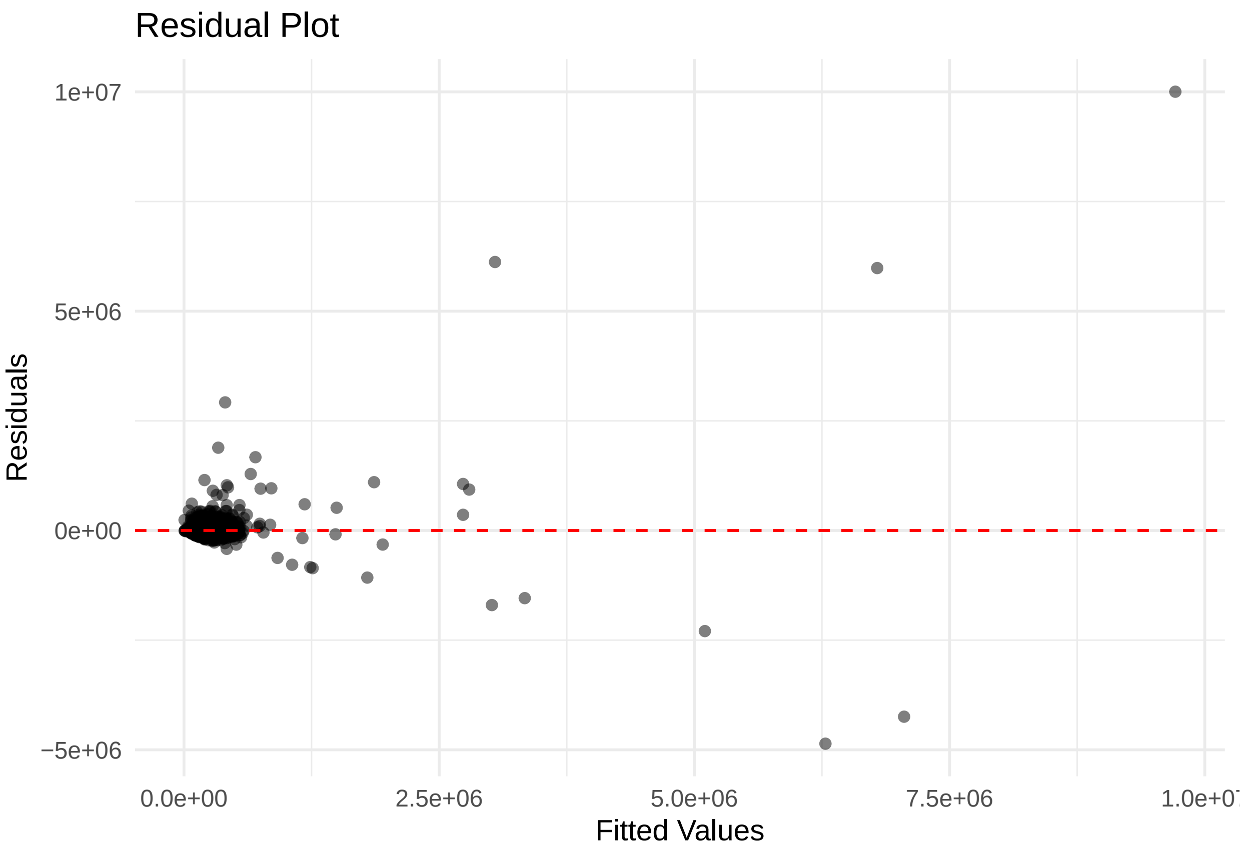


```r
importance_df <- data.frame(
  Feature = importance$var,
  Importance = importance$rel.inf
)

ggplot(importance_df, aes(x = reorder(Feature, Importance), y = Importance)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  coord_flip() +
  theme_minimal() +
  theme(
    plot.title = element_text(face = "bold"),
    axis.text = element_text(size = 10),
    axis.title = element_text(size = 12)
  ) +
  labs(
    x = "Features",
    y = "Relative Importance (%)",
    title = "Feature Importance in House Price Prediction",
    subtitle = paste("Model R² =", round(r2, 3)),
    caption = paste("RMSE: $", format(rmse, big.mark=",", scientific=FALSE))
  )
```



```
# Add residual plot
residuals <- actual_values - predictions
ggplot(data.frame(fitted = predictions, residuals = residuals),
       aes(x = fitted, y = residuals)) +
  geom_point(alpha = 0.5) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  theme_minimal() +
  labs(
    x = "Fitted Values",
    y = "Residuals",
    title = "Residual Plot",
  )
```


Residual Plot
1e+07
5e+06
0e+00
−5e+06
Residuals
0.0e+00
2.5e+06
5.0e+06
7.5e+06
1.0e+07
Fitted Values