```r
###############################################################################
# SDS-301 Modern Regression Analysis
# Final Project: House Price Prediction (Boston Housing)
# Language: R
###############################################################################

# ========================= 0. PACKAGES ======================================

# Install packages once if needed (do NOT run every time)
# install.packages(c("MASS", "tidyverse", "corrplot",
#                    "car", "lmtest", "GGally", "caret"))

library(MASS)       # Boston dataset
library(tidyverse)  # dplyr, ggplot2
library(corrplot)   # correlation plots
library(car)        # VIF
library(lmtest)     # Breusch-Paan test
library(GGally)     # ggpairs
library(caret)      # cross-validation

set.seed(123)

# ========================= 1. LOAD DATA =====================================

data("Boston")
df <- Bostonx``

cat("Number of observations:", nrow(df), "\n")
cat("Number of variables:", ncol(df), "\n\n")

str(df)
summary(df)

# Check missing values
colSums(is.na(df))

# ========================= 2. EDA ===========================================

# ----- 2.1 Descriptive statistics -------------------------------------------
eda_summary <- df %>%
  summarise(across(
    everything(),
    list(
      mean = mean,
      sd   = sd,
      min  = min,
      q1   = ~quantile(.x, 0.25),
      med  = median,
      q3   = ~quantile(.x, 0.75),
      max  = max
    ),
    .names = "{.col}_{.fn}"
  ))
print(eda_summary)

# ----- 2.2 Histograms -------------------------------------------------------
pdf("histograms_all_variables.pdf", width = 10, height = 8)
par(mfrow = c(4, 4), mar = c(3, 3, 2, 1))
for (v in names(df)) {
  hist(df[[v]],
       main = v,
       xlab = "",
```

```r
       col = "lightblue",
       border = "white")
}
dev.off()

# ----- 2.3 Response skewness: log-transformation check ----------------------
hist(df$medv, breaks = 30, col = "lightgray",
     main = "Histogram of MEDV",
     xlab = "Median House Value ($1000s)")

hist(log(df$medv), breaks = 30, col = "lightgray",
     main = "Histogram of log(MEDV)",
     xlab = "log(Median House Value)")

# ----- 2.4 Correlation matrix -----------------------------------------------
cor_mat <- cor(df)
sort(cor_mat[, "medv"], decreasing = TRUE)

pdf("correlation_matrix.pdf", width = 8, height = 8)
corrplot(cor_mat, method = "color", type = "lower",
         tl.col = "black", tl.srt = 45)
dev.off()

# ----- 2.5 Scatterplots with response ---------------------------------------
pdf("pairs_with_medv.pdf", width = 10, height = 10)
ggpairs(df, columns = c("medv","rm","lstat","ptratio","nox","crim","dis","tax"))
dev.off()

# ========================= 3. MODELING ======================================

# Train / test split
train_idx <- createDataPartition(df$medv, p = 0.8, list = FALSE)
train <- df[train_idx, ]
test  <- df[-train_idx, ]

rmse <- function(y, yhat) sqrt(mean((y - yhat)^2))

# ----- 3.1 Full linear model ------------------------------------------------
model_full <- lm(medv ~ ., data = train)
summary(model_full)

# ----- 3.2 Stepwise model ---------------------------------------------------
model_step <- step(model_full, direction = "both", trace = 0)
summary(model_step)

# ----- 3.3 Polynomial model -------------------------------------------------
model_poly <- lm(
  medv ~ rm + I(rm^2) +
    lstat + I(lstat^2) +
    ptratio + nox + chas,
  data = trainhist(df$medv, breaks = 30, probability = TRUE,
                   col = "lightgray", main = "Histogram of MEDV",
                   xlab = "Median House Value ($1000s)")
  lines(density(df$medv), col = "blue", lwd = 2)

)
summary(model_poly)

# ----- 3.4 Log-transformed response model -----------------------------------
```

```
>
> ##############################################################################
> # SDS-301 Modern Regression Analysis
> # Final Project: House Price Prediction (Boston Housing)
> # Language: R
> ##############################################################################
>
> # ========================= 0. PACKAGES ====================================
>
> # Install packages once if needed (do NOT run every time)
> # install.packages(c("MASS", "tidyverse", "corrplot",
> #                    "car", "lmtest", "GGally", "caret"))
>
> library(MASS)       # Boston dataset
> library(tidyverse)  # dplyr, ggplot2
> library(corrplot)   # correlation plots
> library(car)        # VIF
> library(lmtest)     # Breusch-Pagan test
> library(GGally)     # ggpairs
> library(caret)      # cross-validation
>
> set.seed(123)
>
> # ========================= 1. LOAD DATA ===================================
>
> data("Boston")
> df <- Boston
>
> cat("Number of observations:", nrow(df), "\n")
Number of observations: 506
> cat("Number of variables:", ncol(df), "\n\n")
Number of variables: 14

>
> str(df)
'data.frame':   506 obs. of  14 variables:
 $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
 $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
 $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
 $ chas   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
 $ rm     : num  6.58 6.42 7.18 7 7.15 ...
 $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
 $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
 $ rad    : int  1 2 2 3 3 3 5 5 5 5 ...
 $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
 $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
 $ black  : num  397 397 393 395 397 ...
 $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
 $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
> summary(df)
```

```
 3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10   3rd Qu.:0.00000   3rd Qu.:0.6240   3rd Qu.:6.623   3rd Qu.: 94.08
 Max.   :88.97620   Max.   :100.00   Max.   :27.74   Max.   :1.00000   Max.   :0.8710   Max.   :8.780   Max.   :100.00
      dis              rad              tax           ptratio          black           lstat            medv
 Min.   : 1.130   Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   :  0.32   Min.   : 1.73   Min.   : 5.00
 1st Qu.: 2.100   1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.:375.38   1st Qu.: 6.95   1st Qu.:17.02
 Median : 3.207   Median : 5.000   Median :330.0   Median :19.05   Median :391.44   Median :11.36   Median :21.20
 Mean   : 3.795   Mean   : 9.549   Mean   :408.2   Mean   :18.46   Mean   :356.67   Mean   :12.65   Mean   :22.53
 3rd Qu.: 5.188   3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.23   3rd Qu.:16.95   3rd Qu.:25.00
 Max.   :12.127   Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :396.90   Max.   :37.97   Max.   :50.00
>
> # Check missing values
> colSums(is.na(df))
   crim      zn   indus    chas     nox      rm     age     dis     rad     tax ptratio   black   lstat    medv
      0       0       0       0       0       0       0       0       0       0       0       0       0       0
>
> # ======================== 2. EDA =========================================
>
> # ----- 2.1 Descriptive statistics -----------------------------------------
> eda_summary <- df %>%
+     summarise(across(
+         everything(),
+         list(
+             mean = mean,
+             sd   = sd,
+             min  = min,
+             q1   = ~quantile(.x, 0.25),
+             med  = median,
+             q3   = ~quantile(.x, 0.75),
+             max  = max
+         ),
+         .names = "{.col}_{.fn}"
+     ))
> print(eda_summary)
  crim_mean  crim_sd crim_min  crim_q1 crim_med  crim_q3 crim_max  zn_mean    zn_sd zn_min zn_q1 zn_med zn_q3 zn_max
1  3.613524 8.601545  0.00632 0.082045  0.25651 3.677083  88.9762 11.36364 23.32245      0     0      0  12.5    100
  indus_mean indus_sd indus_min indus_q1 indus_med indus_q3 indus_max  chas_mean  chas_sd chas_min chas_q1 chas_med
1   11.13678 6.860353      0.46     5.19      9.69     18.1     27.74 0.06916996 0.253994        0       0        0
  chas_q3 chas_max  nox_mean    nox_sd nox_min nox_q1 nox_med nox_q3 nox_max  rm_mean     rm_sd rm_min  rm_q1 rm_med
1       0        1 0.5546951 0.1158777   0.385  0.449   0.538  0.624   0.871 6.284634 0.7026171  3.561 5.8855 6.2085
   rm_q3 rm_max age_mean   age_sd age_min age_q1 age_med age_q3 age_max dis_mean  dis_sd dis_min   dis_q1 dis_med
1 6.6235   8.78 68.5749 28.14886     2.9 45.025    77.5 94.075     100 3.795043 2.10571  1.1296 2.100175 3.20745
    dis_q3 dis_max rad_mean   rad_sd rad_min rad_q1 rad_med rad_q3 rad_max tax_mean   tax_sd tax_min tax_q1 tax_med
1 5.188425 12.1265 9.549407 8.707259       1      4       5     24      24 408.2372 168.5371     187    279     330
  tax_q3 tax_max ptratio_mean ptratio_sd ptratio_min ptratio_q1 ptratio_med ptratio_q3 ptratio_max black_mean black_sd
1    666     711     18.45553   2.164946        12.6       17.4       19.05       20.2          22    356.674 91.29486
  black_min black_q1 black_med black_q3 black_max lstat_mean lstat_sd lstat_min lstat_q1 lstat_med lstat_q3 lstat_max
1      0.32 375.3775    391.44  396.225     396.9   12.65306 7.141062      1.73     6.95     11.36   16.955     37.97
  medv_mean  medv_sd medv_min medv_q1 medv_med medv_q3 medv_max
1  22.53281 9.197104        5  17.025     21.2      25       50
>
> # ----- 2.2 Histograms ----------------------------------------------------
> pdf("histograms_all_variables.pdf", width = 10, height = 8)
> par(mfrow = c(4, 4), mar = c(3, 3, 2, 1))
> for (v in names(df)) {
+     hist(df[[v]],
+          main = v,
+          xlab = "",
+          col = "lightblue",
```

```
+           border = "white")
+ }
> dev.off()
RStudioGD
        2
>
> # ----- 2.3 Response skewness: log-transformation check ---------------------
> hist(df$medv, breaks = 30, col = "lightgray",
+      main = "Histogram of MEDV",
+      xlab = "Median House Value ($1000s)")
>
> hist(log(df$medv), breaks = 30, col = "lightgray",
+      main = "Histogram of log(MEDV)",
+      xlab = "log(Median House Value)")
>
> # ----- 2.4 Correlation matrix ----------------------------------------------
> cor_mat <- cor(df)
> sort(cor_mat[, "medv"], decreasing = TRUE)
      medv         rm         zn      black        dis       chas        age        rad       crim        nox        tax 
 1.0000000  0.6953599  0.3604453  0.3334608  0.2499287  0.1752602 -0.3769546 -0.3816262 -0.3883046 -0.4273208 -0.4685359 
     indus    ptratio      lstat 
-0.4837252 -0.5077867 -0.7376627 
>
> pdf("correlation_matrix.pdf", width = 8, height = 8)
> corrplot(cor_mat, method = "color", type = "lower",
+          tl.col = "black", tl.srt = 45)
> dev.off()
RStudioGD
        2
>
> # ----- 2.5 Scatterplots with response ---------------------------------------
> pdf("pairs_with_medv.pdf", width = 10, height = 10)
> ggpairs(df, columns = c("medv","rm","lstat","ptratio","nox","crim","dis","tax"))
```

```
> dev.off()
```

```
RStudioGD
        2
>
> # ======================== 3. MODELING ====================================
>
> # Train / test split
> train_idx <- createDataPartition(df$medv, p = 0.8, list = FALSE)
> train <- df[train_idx, ]
> test  <- df[-train_idx, ]
>
> rmse <- function(y, yhat) sqrt(mean((y - yhat)^2))
>
> # ----- 3.1 Full linear model -----------------------------------------------
> model_full <- lm(medv ~ ., data = train)
> summary(model_full)

Call:
lm(formula = medv ~ ., data = train)
```

```
Residuals:
     Min       1Q   Median       3Q      Max 
-14.9550  -2.7996  -0.4647   1.7767  25.0993 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)  37.733617   5.619935   6.714 6.63e-11 ***
crim         -0.093857   0.039157  -2.397 0.016999 *  
zn            0.039436   0.015987   2.467 0.014062 *  
indus        -0.012988   0.069595  -0.187 0.852059    
chas          2.290187   0.940621   2.435 0.015346 *  
nox         -17.130560   4.342272  -3.945 9.45e-05 ***
rm            3.499219   0.451445   7.751 7.87e-14 ***
age           0.009823   0.015510   0.633 0.526905    
dis          -1.390769   0.230614  -6.031 3.77e-09 ***
rad           0.330939   0.077135   4.290 2.25e-05 ***
tax          -0.012386   0.004342  -2.852 0.004568 ** 
ptratio      -0.960676   0.150307  -6.391 4.66e-10 ***
black         0.009841   0.002935   3.353 0.000877 ***
lstat        -0.562095   0.059180  -9.498  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.801 on 393 degrees of freedom
Multiple R-squared:  0.7346,	Adjusted R-squared:  0.7258 
F-statistic: 83.68 on 13 and 393 DF,  p-value: < 2.2e-16

> 
> # ----- 3.2 Stepwise model --------------------------------------------------
> model_step <- step(model_full, direction = "both", trace = 0)
> summary(model_step)

Call:
lm(formula = medv ~ crim + zn + chas + nox + rm + dis + rad + 
    tax + ptratio + black + lstat, data = train)

Residuals:
     Min       1Q   Median       3Q      Max 
-15.0826  -2.7796  -0.4605   1.7818  25.4524 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)  37.517928   5.576559   6.728 6.07e-11 ***
crim         -0.094346   0.039042  -2.416 0.016123 *  
zn            0.038552   0.015811   2.438 0.015195 *  
chas          2.289157   0.932318   2.455 0.014505 *  
nox         -16.651928   4.010077  -4.153 4.03e-05 ***
rm            3.569475   0.437455   8.160 4.54e-15 ***
dis          -1.425853   0.213439  -6.680 8.12e-11 ***
rad           0.330725   0.074617   4.432 1.21e-05 ***
tax          -0.012607   0.004010  -3.144 0.001793 ** 
ptratio      -0.957908   0.147191  -6.508 2.31e-10 ***
black         0.010003   0.002919   3.427 0.000674 ***
lstat        -0.548693   0.054617 -10.046  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.791 on 395 degrees of freedom
Multiple R-squared:  0.7343,	Adjusted R-squared:  0.7269 
F-statistic: 99.24 on 11 and 395 DF,  p-value: < 2.2e-16

> 
> # ----- 3.3 Polynomial model ------------------------------------------------
> model_poly <- lm(
+     medv ~ rm + I(rm^2) +
+         lstat + I(lstat^2) +
+         ptratio + nox + chas,
+     data = train
+ )
> summary(model_poly)

Call:
lm(formula = medv ~ rm + I(rm^2) + lstat + I(lstat^2) + ptratio + 
    nox + chas, data = train)

Residuals:
     Min       1Q   Median       3Q      Max 
-27.2895  -2.3606  -0.3948   2.2084  28.7068 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) 112.172797  10.298306  10.892  < 2e-16 ***
rm          -24.185133   3.134306  -7.716 9.70e-14 ***
I(rm^2)       2.170092   0.243956   8.895  < 2e-16 ***
lstat        -1.153391   0.141278  -8.164 4.30e-15 ***
I(lstat^2)    0.016183   0.003836   4.219 3.04e-05 ***
ptratio      -0.621126   0.114000  -5.448 8.91e-08 ***
nox          -3.685324   2.459269  -1.499 0.134782    
chas          2.914602   0.845295   3.448 0.000625 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.388 on 399 degrees of freedom
Multiple R-squared:  0.7749,	Adjusted R-squared:  0.7709 
F-statistic: 196.2 on 7 and 399 DF,  p-value: < 2.2e-16

> 
> # ----- 3.4 Log-transformed response model ----------------------------------
> model_log <- lm(log(medv) ~ ., data = train)
> summary(model_log)

Call:
lm(formula = log(medv) ~ ., data = train)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.75651 -0.09795 -0.01562  0.09365  0.87472 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)  4.0742251  0.2194337  18.567  < 2e-16 ***
crim        -0.0086881  0.0015289  -5.683 2.59e-08 ***
zn           0.0010046  0.0006242   1.609 0.108363    
indus        0.0016972  0.0027174   0.625 0.532606    
```

```
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  4.0742251  0.2194337  18.567  < 2e-16 ***
crim        -0.0086881  0.0015289  -5.683 2.59e-08 ***
zn           0.0010046  0.0006242   1.609 0.108363
indus        0.0016972  0.0027174   0.625 0.532606
chas         0.0922525  0.0367271   2.512 0.012410 *
nox         -0.7689606  0.1695466  -4.535 7.65e-06 ***
rm           0.0822323  0.0176269   4.665 4.23e-06 ***
age          0.0007802  0.0006056   1.288 0.198424
dis         -0.0443855  0.0090044  -4.929 1.22e-06 ***
rad          0.0141637  0.0030118   4.703 3.56e-06 ***
tax         -0.0005809  0.0001695  -3.427 0.000676 ***
ptratio     -0.0381756  0.0058688  -6.505 2.37e-10 ***
black        0.0004942  0.0001146   4.313 2.04e-05 ***
lstat       -0.0307852  0.0023107 -13.323  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1874 on 393 degrees of freedom
Multiple R-squared:  0.7918,	Adjusted R-squared:  0.7849 
F-statistic:   115 on 13 and 393 DF,  p-value: < 2.2e-16

> 
> # ======================== 4. DIAGNOSTICS & SELECTION ======================
> 
> # Diagnostics for stepwise model
> par(mfrow = c(2, 2))
> plot(model_step)
> par(mfrow = c(1, 1))
> 
> # Residual tests
> shapiro.test(residuals(model_step))

	Shapiro-Wilk normality test

data:  residuals(model_step)
W = 0.8981, p-value = 7.515e-16

> bptest(model_step)

	studentized Breusch-Pagan test

data:  model_step
BP = 52.044, df = 11, p-value = 2.677e-07

> vif(model_step)
    crim       zn     chas      nox       rm      dis      rad      tax  ptratio    black    lstat 
1.841093 2.281268 1.052281 3.808416 1.802376 3.496271 7.659422 8.271439 1.782706 1.354930 2.678063 
> 
> # Diagnostics for log model
> par(mfrow = c(2, 2))
> plot(model_log)
> par(mfrow = c(1, 1))
> 
> shapiro.test(residuals(model_log))

	Shapiro-Wilk normality test
```

```
        Shapiro-Wilk normality test

data:  residuals(model_log)
W = 0.95164, p-value = 2.793e-10

> bptest(model_log)

        studentized Breusch-Pagan test

data:  model_log
BP = 52.947, df = 13, p-value = 9.23e-07

> vif(model_log)
    crim       zn    indus     chas      nox       rm      age      dis      rad      tax  ptratio    black    lstat 
1.844585 2.323334 3.956419 1.066866 4.447852 1.911905 3.213786 4.065420 8.152643 9.660128 1.851645 1.364539 3.131835 
> 
> # ========================= 5. MODEL COMPARISON ==============================
> 
> # Predictions on test set
> pred_full  <- predict(model_full, newdata = test)
> pred_step  <- predict(model_step, newdata = test)
> pred_poly  <- predict(model_poly, newdata = test)
> pred_log   <- exp(predict(model_log, newdata = test))  # back-transform
> 
> model_comp <- data.frame(
+     Model = c("Full", "Stepwise", "Polynomial", "Log-Linear"),
+     RMSE  = c(rmse(test$medv, pred_full),
+               rmse(test$medv, pred_step),
+               rmse(test$medv, pred_poly),
+               rmse(test$medv, pred_log)),
+     R2    = c(cor(test$medv, pred_full)^2,
+               cor(test$medv, pred_step)^2,
+               cor(test$medv, pred_poly)^2,
+               cor(test$medv, pred_log)^2)
+ )
> print(model_comp)
       Model     RMSE        R2
1       Full 4.588948 0.7611260
2   Stepwise 4.560599 0.7646440
3 Polynomial 4.289833 0.7958492
4 Log-Linear 4.208077 0.8105357
> 
> # Cross-validation for stepwise model
> train_control <- trainControl(method = "cv", number = 10)
> cv_step <- train(
+     medv ~ .,
+     data = train[, all.vars(formula(model_step))],
+     method = "lm",
+     trControl = train_control
+ )
> cv_step$results
  intercept     RMSE  Rsquared      MAE   RMSESD RsquaredSD     MAESD
1      TRUE 4.794382 0.7350305 3.391633 1.181335  0.1150608 0.7162708
> 
> # ========================= 6. FINAL MODEL ==================================
> 
> # Choosing log-linear as final model for better residuals
> final_model <- model_log
```

```
Call:
lm(formula = log(medv) ~ ., data = train)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.75651 -0.09795 -0.01562  0.09365  0.87472 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)  4.0742251  0.2194337  18.567  < 2e-16 ***
crim        -0.0086881  0.0015289  -5.683 2.59e-08 ***
zn           0.0010046  0.0006242   1.609 0.108363    
indus        0.0016972  0.0027174   0.625 0.532606    
chas         0.0922525  0.0367271   2.512 0.012410 *  
nox         -0.7689606  0.1695466  -4.535 7.65e-06 ***
rm           0.0822323  0.0176269   4.665 4.23e-06 ***
age          0.0007802  0.0006056   1.288 0.198424    
dis         -0.0443855  0.0090044  -4.929 1.22e-06 ***
rad          0.0141637  0.0030118   4.703 3.56e-06 ***
tax         -0.0005809  0.0001695  -3.427 0.000676 ***
ptratio     -0.0381756  0.0058688  -6.505 2.37e-10 ***
black        0.0004942  0.0001146   4.313 2.04e-05 ***
lstat       -0.0307852  0.0023107 -13.323  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1874 on 393 degrees of freedom
Multiple R-squared:  0.7918,	Adjusted R-squared:  0.7849 
F-statistic:   115 on 13 and 393 DF,  p-value: < 2.2e-16

> 
> # Predictions and metrics
> final_train_pred <- exp(predict(final_model, train))
> final_test_pred  <- exp(predict(final_model, test))
> 
> final_results <- data.frame(
+     Set  = c("Train", "Test"),
+     R2   = c(cor(train$medv, final_train_pred)^2,
+              cor(test$medv, final_test_pred)^2),
+     RMSE = c(rmse(train$medv, final_train_pred),
+              rmse(test$medv, final_test_pred))
+ )
> print(final_results)
    Set        R2     RMSE
1 Train 0.7824145 4.331008
2  Test 0.8105357 4.208077
> 
> # ========================= 7. INTERPRETATION ================================
> 
> coefs <- coef(final_model)
> 
> cat("\nLog-linear model interpretation examples:\n")

Log-linear model interpretation examples:
```

```r
# ----- 3.4 Log-transformed response model -----------------------------------
model_log <- lm(log(medv) ~ ., data = train)
summary(model_log)

# ========================= 4. DIAGNOSTICS & SELECTION =======================

# Diagnostics for stepwise model
par(mfrow = c(2, 2))
plot(model_step)
par(mfrow = c(1, 1))

# Residual tests
shapiro.test(residuals(model_step))
bptest(model_step)
vif(model_step)

# Diagnostics for log model
par(mfrow = c(2, 2))
plot(model_log)
par(mfrow = c(1, 1))

shapiro.test(residuals(model_log))
bptest(model_log)
vif(model_log)

# ========================= 5. MODEL COMPARISON ================================

# Predictions on test set
pred_full   <- predict(model_full, newdata = test)
pred_step   <- predict(model_step, newdata = test)
pred_poly   <- predict(model_poly, newdata = test)
pred_log    <- exp(predict(model_log, newdata = test))  # back-transform

model_comp <- data.frame(
  Model  = c("Full", "Stepwise", "Polynomial", "Log-Linear"),
  RMSE   = c(rmse(test$medv, pred_full),
             rmse(test$medv, pred_step),
             rmse(test$medv, pred_poly),
             rmse(test$medv, pred_log)),
  R2     = c(cor(test$medv, pred_full)^2,
             cor(test$medv, pred_step)^2,
             cor(test$medv, pred_poly)^2,
             cor(test$medv, pred_log)^2)
)
print(model_comp)

# Cross-validation for stepwise model
train_control <- trainControl(method = "cv", number = 10)
cv_step <- train(
  medv ~ .,
  data = train[, all.vars(formula(model_step))],
  method = "lm",
  trControl = train_control
)
cv_step$results
```

```r
# ========================= 6. FINAL MODEL ===================================

# Choosing log-linear as final model for better residuals
final_model <- model_log
summary(final_model)

# Predictions and metrics
final_train_pred <- exp(predict(final_model, train))
final_test_pred  <- exp(predict(final_model, test))

final_results <- data.frame(
  Set  = c("Train", "Test"),
  R2   = c(cor(train$medv, final_train_pred)^2,
           cor(test$medv, final_test_pred)^2),
  RMSE = c(rmse(train$medv, final_train_pred),
           rmse(test$medv, final_test_pred))
)
print(final_results)

# ========================= 7. INTERPRETATION ================================

coefs <- coef(final_model)

cat("\nLog-linear model interpretation examples:\n")
cat("rm: a one-unit increase in rooms increases median house value by approximately",
    round((exp(coefs["rm"]) - 1) * 100, 2), "%, holding other variables constant.\n")
cat("lstat: a 1% increase in lower-status population decreases median house value by approximately",
    round((exp(coefs["lstat"]) - 1) * 100, 2), "%.\n")

# ========================= 8. SAVE OUTPUTS ==================================

coef_df <- as.data.frame(summary(final_model)$coefficients)
coef_df$Variable <- rownames(coef_df)
coef_df <- coef_df[, c("Variable", "Estimate", "Std. Error", "t value", "Pr(>|t|)")]

write.csv(coef_df, "final_model_coefficients.csv", row.names = FALSE)
write.csv(final_results, "final_model_metrics.csv", row.names = FALSE)

cat("\n===== DONE: Code executed successfully. All outputs saved. =====\n")
##############################################################################
```