STA 141C GBM

Rohan Khubchandani

2023-05-19

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com (http://rmarkdown.rstudio.com).

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
library(gbm)
```

```
## Loaded gbm 2.2.2
```

```
## This version of gbm is no longer under development. Consider transitioning to gbm3, https://github.com/gbm-dev
elopers/gbm3
```

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
# Load data
train <- read.csv("/Users/rohankhubchandani/Desktop/train.csv")

# Add log1p-transformed SalePrice for log-scale modeling
train$LogSalePrice <- log1p(train$SalePrice)

# Drop ID and original SalePrice (model will use LogSalePrice)
train$Id <- NULL
train$SalePrice <- NULL

# Remove columns with too many NAs (e.g., PoolQC, MiscFeature, Alley)
train <- train[, colMeans(!is.na(train)) > 0.8]

# Define Mode function
Mode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}

# Impute missing values
for (col in names(train)) {
  if (any(is.na(train[[col]]))) {
    if (is.numeric(train[[col]])) {
      train[[col]][is.na(train[[col]])] <- median(train[[col]], na.rm = TRUE)
    } else {
      train[[col]][is.na(train[[col]])] <- Mode(train[[col]])
    }
  }
}



# Convert character columns to factors
for (col in names(train)) {
  if (is.character(train[[col]])) {
    train[[col]] <- as.factor(train[[col]])
  }
}

numeric_cols <- sapply(train, is.numeric)
numeric_data <- train[, numeric_cols]

skewness <- function(x) {
  n <- length(x)
  m2 <- mean((x - mean(x))^2)
```

```r
  m3 <- mean((x - mean(x))^3)
  m3 / (m2 ^ 1.5)
}

for (col in names(numeric_data)) {
  skew_val <- skewness(train[[col]])
  if (skew_val > 1) {
    train[[col]] <- log1p(train[[col]])
  }
}

for (col in names(train)[numeric_cols]) {
  p_low <- quantile(train[[col]], probs = 0.005, na.rm = TRUE)
  p_high <- quantile(train[[col]], probs = 0.95, na.rm = TRUE)

  train[[col]][train[[col]] < p_low] <- p_low
  train[[col]][train[[col]] > p_high] <- p_high
}

low_var_cols <- c()
for (col in names(train)[numeric_cols]) {
  if (length(unique(train[[col]])) == 1) {
    low_var_cols <- c(low_var_cols, col)
  } else {
    ratio <- var(train[[col]]) / (mean(train[[col]])^2)
    if (!is.na(ratio) && ratio < 0.01) {
      low_var_cols <- c(low_var_cols, col)
    }
  }
}

# Fit GBM model on log-transformed SalePrice
set.seed(123)
gbm_model <- gbm(LogSalePrice ~ .,
                 data = train,
                 distribution = "gaussian",
                 n.trees = 1000,
                 interaction.depth = 4,
                 shrinkage = 0.01,
                 cv.folds = 5,
                 n.cores = 4,
                 verbose = FALSE)
```

```
## Warning in gbm.fit(x = x, y = y, offset = offset, distribution = distribution,
## : variable 44: LowQualFinSF has no variation.
```

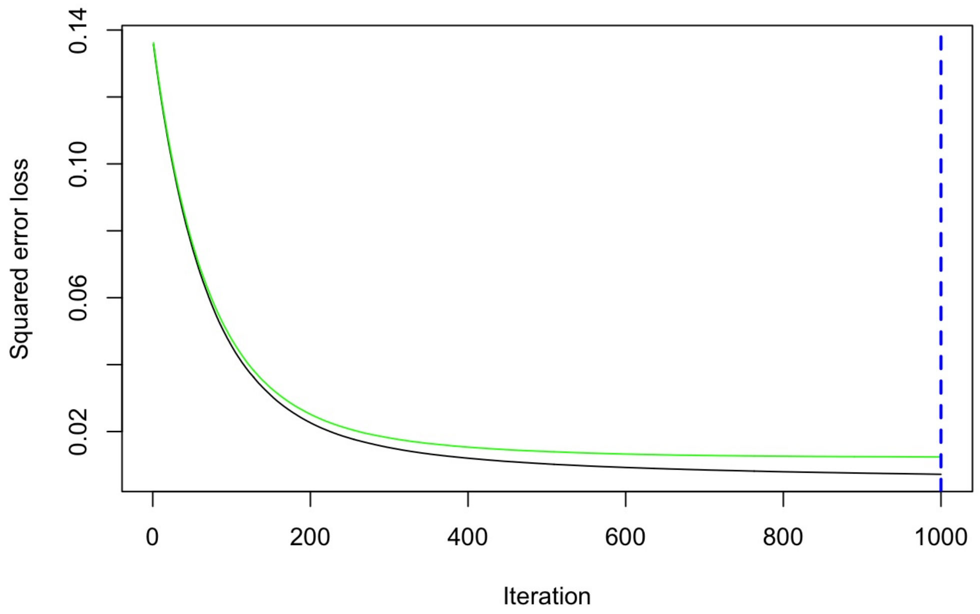
```
## Warning in gbm.fit(x = x, y = y, offset = offset, distribution = distribution,
## : variable 51: KitchenAbvGr has no variation.
```

```
## Warning in gbm.fit(x = x, y = y, offset = offset, distribution = distribution,
## : variable 67: X3SsnPorch has no variation.
```

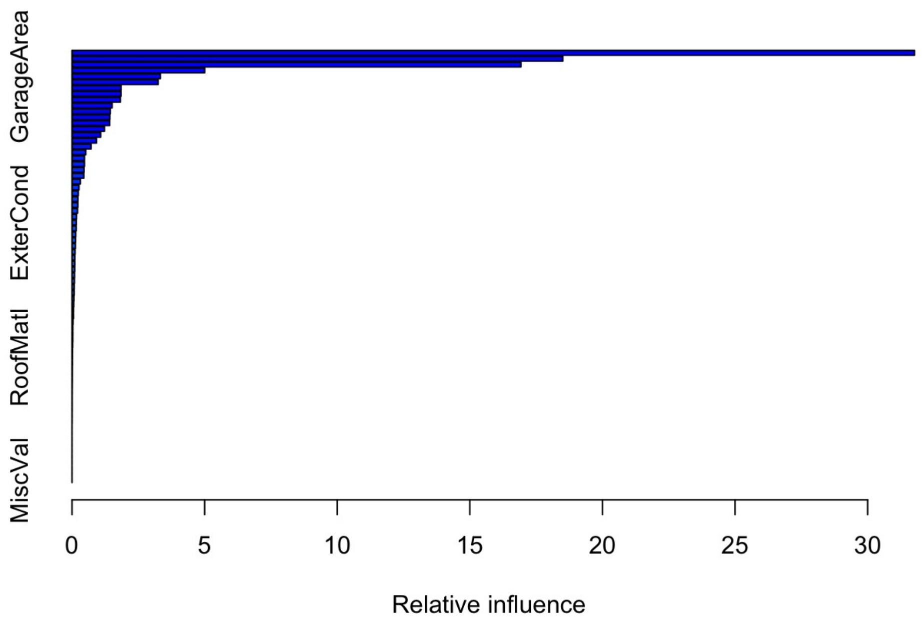
```
## Warning in gbm.fit(x = x, y = y, offset = offset, distribution = distribution,
## : variable 69: PoolArea has no variation.
```

```
## Warning in gbm.fit(x = x, y = y, offset = offset, distribution = distribution,
## : variable 70: MiscVal has no variation.
```

```r
# Optimal number of trees based on CV
best_iter <- gbm.perf(gbm_model, method = "cv")
```




```r
# Variable importance
summary(gbm_model, n.trees = best_iter)
```



```
##                           var      rel.inf
## OverallQual     OverallQual 31.765281553
## Neighborhood   Neighborhood 18.506920991
## GrLivArea         GrLivArea 16.925568931
## TotalBsmtSF     TotalBsmtSF  5.000997002
## GarageArea       GarageArea  3.332518540
## KitchenQual     KitchenQual  3.245025668
## OverallCond     OverallCond  1.847560573
## BsmtFinSF1       BsmtFinSF1  1.846005478
## X1stFlrSF         X1stFlrSF  1.823645591
## GarageCars       GarageCars  1.511293453
## GarageFinish   GarageFinish  1.443442573
## CentralAir       CentralAir  1.425860238
## YearRemodAdd   YearRemodAdd  1.418442694
## LotArea             LotArea  1.222775858
## ExterQual         ExterQual  1.078692385
## Fireplaces       Fireplaces  0.922788845
## SaleCondition SaleCondition  0.716577711
## YearBuilt         YearBuilt  0.518657281
## Exterior1st     Exterior1st  0.468548587
## BsmtFinType1   BsmtFinType1  0.468181501
## Functional       Functional  0.450734060
## X2ndFlrSF         X2ndFlrSF  0.444054606
## BsmtQual           BsmtQual  0.315101585
## WoodDeckSF       WoodDeckSF  0.261933084
## Exterior2nd     Exterior2nd  0.234258242
## Condition1       Condition1  0.218564936
## GarageType       GarageType  0.216270312
## BsmtExposure   BsmtExposure  0.207154010
## PavedDrive       PavedDrive  0.168579573
## ExterCond         ExterCond  0.160399376
## OpenPorchSF     OpenPorchSF  0.157311951
## BsmtCond           BsmtCond  0.139922439
## BsmtFullBath   BsmtFullBath  0.132237802
## ScreenPorch     ScreenPorch  0.126783070
## MSZoning           MSZoning  0.113849648
## GarageYrBlt     GarageYrBlt  0.104933859
## HalfBath           HalfBath  0.102856361
## HeatingQC         HeatingQC  0.100500949
## LandContour     LandContour  0.093679432
## YrSold               YrSold  0.089918356
## BsmtUnfSF         BsmtUnfSF  0.082102366
## LotFrontage     LotFrontage  0.080559275
## MoSold               MoSold  0.069988559
## BldgType           BldgType  0.063377153
## SaleType           SaleType  0.055362307
## HouseStyle       HouseStyle  0.055264577
## Foundation       Foundation  0.037557595
## RoofStyle         RoofStyle  0.029607289
## BedroomAbvGr   BedroomAbvGr  0.029261041
## LotConfig         LotConfig  0.028405460
## EnclosedPorch EnclosedPorch  0.023378464
## MasVnrArea       MasVnrArea  0.016892055
## RoofMatl           RoofMatl  0.015523630
## GarageQual       GarageQual  0.014938247
## MSSubClass       MSSubClass  0.012077225
## Condition2       Condition2  0.011289398
## GarageCond       GarageCond  0.009850718
## TotRmsAbvGrd   TotRmsAbvGrd  0.007995407
## FullBath           FullBath  0.006205319
## BsmtFinType2   BsmtFinType2  0.005295242
## LotShape           LotShape  0.005221701
## Electrical       Electrical  0.004936606
## LandSlope         LandSlope  0.003965765
## BsmtHalfBath   BsmtHalfBath  0.003115497
## Street               Street  0.000000000
## Utilities         Utilities  0.000000000
## MasVnrType       MasVnrType  0.000000000
## BsmtFinSF2       BsmtFinSF2  0.000000000
## Heating             Heating  0.000000000
## LowQualFinSF   LowQualFinSF  0.000000000
## KitchenAbvGr   KitchenAbvGr  0.000000000
## X3SsnPorch       X3SsnPorch  0.000000000
## PoolArea           PoolArea  0.000000000
## MiscVal             MiscVal  0.000000000
```

```r
# Predict in log space
log_preds <- predict(gbm_model, newdata = train, n.trees = best_iter)

# Compute RMSE in log space
log_actuals <- train$LogSalePrice
rmse_log <- sqrt(mean((log_preds - log_actuals)^2))
cat(" Log-scale RMSE:", round(rmse_log, 4), "\n")
```

```
##  Log-scale RMSE: 0.085
```

```r
# Convert predictions back to original price scale
preds <- expm1(log_preds)
actuals <- expm1(log_actuals)

# RMSE in actual dollars
rmse_dollar <- sqrt(mean((preds - actuals)^2))
cat(" Dollar-scale RMSE:", round(rmse_dollar, 2), "\n")
```

```
##  Dollar-scale RMSE: 14544.09
```

```r
# Create submission file with log-scale predictions
submission_train_log <- data.frame(
  LogPrediction = log_preds,
  ActualLogSalePrice = log_actuals,
  LogResidual = log_preds - log_actuals
)

write.csv(submission_train_log, "submission_train_log.csv", row.names = FALSE)
cat(" Log-scale predictions saved to 'submission_train_log.csv'\n")
```

```
##  Log-scale predictions saved to 'submission_train_log.csv'
```

```r
library(gbm)
library(caret)

# ------------------ Load and Prepare Training Data ------------------
train <- read.csv("/Users/rohankhubchandani/Desktop/train.csv")

# Save original SalePrice for later RMSE evaluation
train$SalePrice_Original <- train$SalePrice

# Log-transform the target
train$LogSalePrice <- log1p(train$SalePrice)

# Drop original SalePrice and ID
train$SalePrice <- NULL
train$Id <- NULL

# Remove columns with too many missing values
train <- train[, colMeans(!is.na(train)) > 0.8]

# Mode function for categorical imputation
Mode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}

# Impute missing values
for (col in names(train)) {
  if (any(is.na(train[[col]]))) {
    if (is.numeric(train[[col]])) {
      train[[col]][is.na(train[[col]])] <- median(train[[col]], na.rm = TRUE)
    } else {
      train[[col]][is.na(train[[col]])] <- Mode(train[[col]])
    }
  }
}

# Convert character columns to factors
for (col in names(train)) {
  if (is.character(train[[col]])) {
    train[[col]] <- as.factor(train[[col]])
  }
}
```

```r
numeric_cols <- sapply(train, is.numeric)
numeric_data <- train[, numeric_cols]

skewness <- function(x) {
  n <- length(x)
  m2 <- mean((x - mean(x))^2)
  m3 <- mean((x - mean(x))^3)
  m3 / (m2 ^ 1.5)
}

for (col in names(numeric_data)) {
  skew_val <- skewness(train[[col]])
  if (skew_val > 1) {
    train[[col]] <- log1p(train[[col]])
  }
}

for (col in names(train)[numeric_cols]) {
  p_low <- quantile(train[[col]], probs = 0.005, na.rm = TRUE)
  p_high <- quantile(train[[col]], probs = 0.95, na.rm = TRUE)

  train[[col]][train[[col]] < p_low] <- p_low
  train[[col]][train[[col]] > p_high] <- p_high
}

low_var_cols <- c()
for (col in names(train)[numeric_cols]) {
  if (length(unique(train[[col]])) == 1) {
    low_var_cols <- c(low_var_cols, col)
  } else {
    ratio <- var(train[[col]]) / (mean(train[[col]])^2)
    if (!is.na(ratio) && ratio < 0.01) {
      low_var_cols <- c(low_var_cols, col)
    }
  }
}



# ------------------ Train/Validation Split ------------------
set.seed(123)
train_index <- createDataPartition(train$LogSalePrice, p = 0.8, list = FALSE)
train_data <- train[train_index, ]
val_data <- train[-train_index, ]

# ------------------ Train GBM ------------------
gbm_model <- gbm(LogSalePrice ~ . -SalePrice_Original,
                 data = train_data,
                 distribution = "gaussian",
                 n.trees = 1000,
                 interaction.depth = 4,
                 shrinkage = 0.01,
                 cv.folds = 5,
                 verbose = FALSE)
```

```
## Warning in gbm.fit(x = x, y = y, offset = offset, distribution = distribution,
## : variable 44: LowQualFinSF has no variation.
```

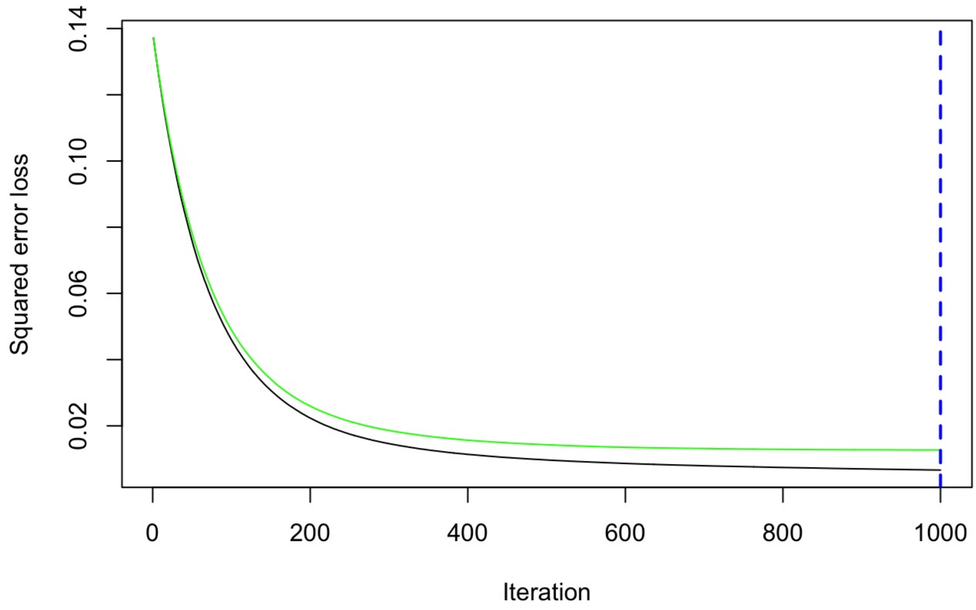
```
## Warning in gbm.fit(x = x, y = y, offset = offset, distribution = distribution,
## : variable 51: KitchenAbvGr has no variation.
```

```
## Warning in gbm.fit(x = x, y = y, offset = offset, distribution = distribution,
## : variable 67: X3SsnPorch has no variation.
```

```
## Warning in gbm.fit(x = x, y = y, offset = offset, distribution = distribution,
## : variable 69: PoolArea has no variation.
```

```
## Warning in gbm.fit(x = x, y = y, offset = offset, distribution = distribution,
## : variable 70: MiscVal has no variation.
```

```r
best_iter <- gbm.perf(gbm_model, method = "cv")
```




```
# ------------------ Evaluate on Validation Set ------------------
log_preds_val <- predict(gbm_model, newdata = val_data, n.trees = best_iter)
log_actuals_val <- val_data$LogSalePrice

# Log-scale RMSE
rmse_log <- sqrt(mean((log_preds_val - log_actuals_val)^2))
cat(" Log-scale RMSE (Validation):", round(rmse_log, 4), "\n")
```

```
##  Log-scale RMSE (Validation): 0.1111
```

```
# Percent error
percent_error <- (exp(rmse_log) - 1) * 100
cat(" Approximate Percent Error:", round(percent_error, 2), "%\n")
```

```
##  Approximate Percent Error: 11.75 %
```

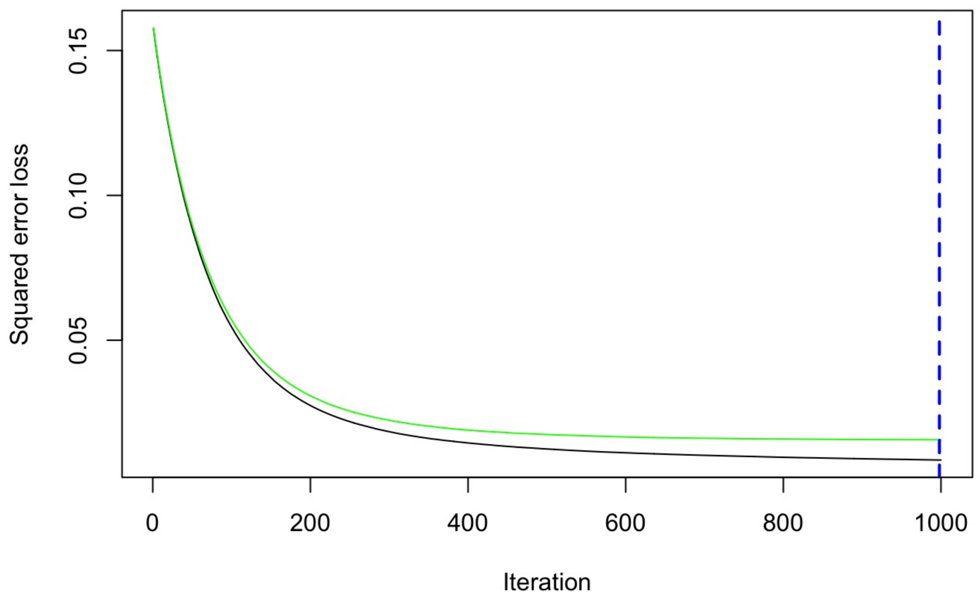
```
# Dollar-scale RMSE
preds_dollar <- expm1(log_preds_val)
actuals_dollar <- val_data$SalePrice_Original
rmse_dollar <- sqrt(mean((preds_dollar - actuals_dollar)^2))
cat(" Dollar-scale RMSE (Validation):", round(rmse_dollar, 2), "\n")
```

```
##  Dollar-scale RMSE (Validation): 188456.6
```

```r
# ------------------ Load and Prepare Test Data ------------------
test <- read.csv("/Users/rohankhubchandani/Desktop/test.csv")
test_ids <- test$Id
test$Id <- NULL

# Keep only columns that overlap with training set
test <- test[, colnames(test) %in% colnames(train_data)]

# Impute missing values
for (col in names(test)) {
  if (any(is.na(test[[col]]))) {
    if (is.numeric(test[[col]])) {
      test[[col]][is.na(test[[col]])] <- median(test[[col]], na.rm = TRUE)
    } else {
      test[[col]][is.na(test[[col]])] <- Mode(test[[col]])
    }
  }
}

# Convert characters to factors
for (col in names(test)) {
  if (is.character(test[[col]])) {
    test[[col]] <- as.factor(test[[col]])
  }
}



# ------------------ Predict on Test and Create Submission ------------------
log_preds_test <- predict(gbm_model, newdata = test, n.trees = best_iter)
preds_test <- expm1(log_preds_test)

submission <- data.frame(Id = test_ids, SalePrice = preds_test)
write.csv(submission, "submission.csv", row.names = FALSE)
cat(" Kaggle submission file 'submission.csv' created.\n")
```

```
##  Kaggle submission file 'submission.csv' created.
```

```r
library(gbm)

# Load data
train <- read.csv("/Users/rohankhubchandani/Desktop/train.csv")

# Save original SalePrice for comparison later
train$OriginalSalePrice <- train$SalePrice

# Create log-transformed response
train$LogSalePrice <- log1p(train$SalePrice)

# Drop ID and original SalePrice (keeping OriginalSalePrice for RMSE)
train$Id <- NULL
train$SalePrice <- NULL

# Mode function for imputation
Mode <- function(x) {
  ux <- unique(x[!is.na(x)])
  ux[which.max(tabulate(match(x, ux)))]
}

# Impute missing values
for (col in names(train)) {
  if (any(is.na(train[[col]]))) {
    if (is.numeric(train[[col]])) {
      train[[col]][is.na(train[[col]])] <- median(train[[col]], na.rm = TRUE)
    } else {
      train[[col]][is.na(train[[col]])] <- Mode(train[[col]])
    }
  }
}

# Convert character columns to factors
for (col in names(train)) {
  if (is.character(train[[col]])) {
    train[[col]] <- as.factor(train[[col]])
  }
}

# Fit GBM on log-transformed SalePrice
set.seed(123)
gbm_model <- gbm(LogSalePrice ~ . -OriginalSalePrice,
                 data = train,
                 distribution = "gaussian",
                 n.trees = 1000,
                 interaction.depth = 4,
                 shrinkage = 0.01,
                 cv.folds = 5,
                 n.cores = 4,
                 verbose = FALSE)

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




```
# Predictions in log space
log_preds <- predict(gbm_model, newdata = train, n.trees = best_iter)

# RMSE in log dollars
log_actuals <- train$LogSalePrice
rmse_log <- sqrt(mean((log_preds - log_actuals)^2))
cat(" Log-scale RMSE:", round(rmse_log, 4), "\n")
```

```
##  Log-scale RMSE: 0.093
```

```
# RMSE in original dollars
preds_dollar <- expm1(log_preds)
actuals_dollar <- train$OriginalSalePrice
rmse_dollar <- sqrt(mean((preds_dollar - actuals_dollar)^2))
cat(" Dollar-scale RMSE:", round(rmse_dollar, 2), "\n")
```

```
##  Dollar-scale RMSE: 18804.23
```