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
```

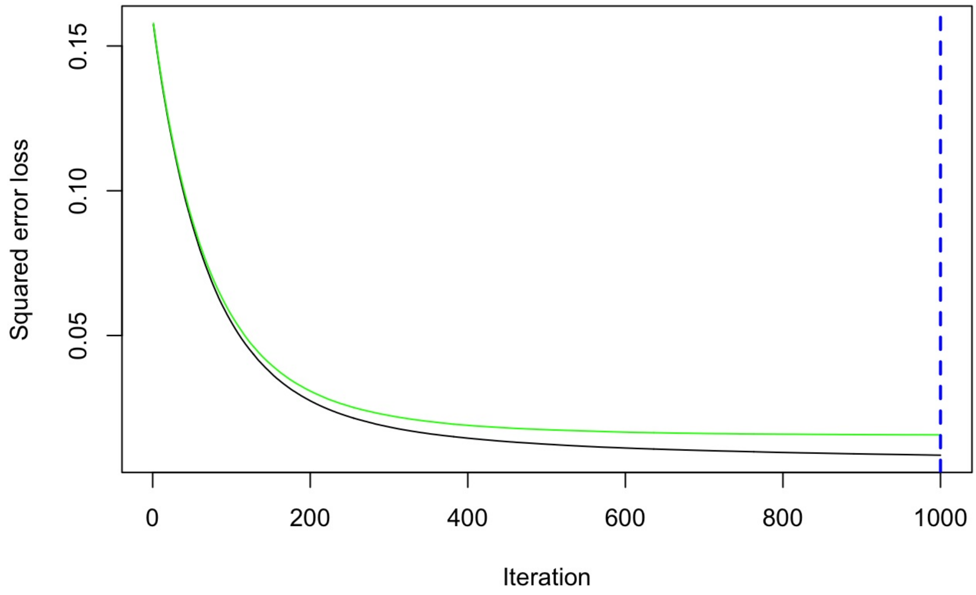
```
## This version of gbm is no longer under development. Consider transitioning to gbm3, https://github.com/gbm-dev
elopers/gbm3
```

```
library(caret)
```

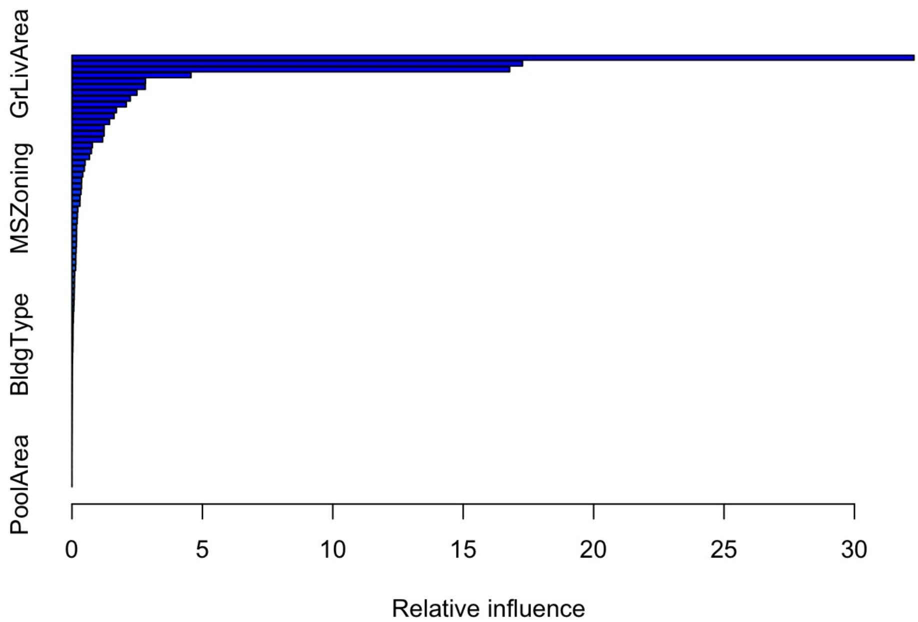
```
## Loading required package: ggplot2
```

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

```r
# 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]])
  }
}

# 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)

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




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



```
##                          var      rel.inf
## OverallQual     OverallQual 32.283483344
## GrLivArea         GrLivArea 17.273856610
## Neighborhood   Neighborhood 16.780864021
## TotalBsmtSF     TotalBsmtSF  4.562710012
## KitchenQual     KitchenQual  2.818968730
## GarageArea       GarageArea  2.810219660
## X1stFlrSF         X1stFlrSF  2.490375821
## BsmtFinSF1       BsmtFinSF1  2.237561483
## GarageCars       GarageCars  2.082739473
## CentralAir       CentralAir  1.704714091
## OverallCond     OverallCond  1.614715561
## ExterQual         ExterQual  1.439020844
## YearRemodAdd   YearRemodAdd  1.234465060
## GarageFinish   GarageFinish  1.231400913
## LotArea             LotArea  1.176148589
## Fireplaces       Fireplaces  0.781756432
## SaleCondition SaleCondition  0.745320779
## BsmtQual           BsmtQual  0.669535572
## YearBuilt         YearBuilt  0.503272693
## X2ndFlrSF         X2ndFlrSF  0.474201708
## BsmtFinType1   BsmtFinType1  0.414480113
## Functional       Functional  0.370705022
## Exterior1st     Exterior1st  0.364698294
## OpenPorchSF     OpenPorchSF  0.346112111
## MSZoning           MSZoning  0.309154682
## BsmtExposure   BsmtExposure  0.304771043
## Exterior2nd     Exterior2nd  0.220885827
## Condition1       Condition1  0.208594040
## LotFrontage     LotFrontage  0.200489443
## ExterCond         ExterCond  0.177419564
## PavedDrive       PavedDrive  0.175831994
## WoodDeckSF       WoodDeckSF  0.174096504
## GarageType       GarageType  0.169129023
## ScreenPorch     ScreenPorch  0.157163403
## HalfBath           HalfBath  0.147062494
## GarageYrBlt     GarageYrBlt  0.146842197
## FullBath           FullBath  0.140757467
## LandContour     LandContour  0.102169010
## HeatingQC         HeatingQC  0.096404752
## TotRmsAbvGrd   TotRmsAbvGrd  0.094565141
## BsmtFullBath   BsmtFullBath  0.087015399
## YrSold               YrSold  0.084620347
## SaleType           SaleType  0.075038503
## RoofMatl           RoofMatl  0.068630964
## BsmtCond           BsmtCond  0.053849530
## BsmtUnfSF         BsmtUnfSF  0.051201477
## MoSold               MoSold  0.039052241
## Foundation       Foundation  0.036977584
## LotConfig         LotConfig  0.036533136
## BldgType           BldgType  0.033336494
## EnclosedPorch EnclosedPorch  0.029504929
## HouseStyle       HouseStyle  0.021832005
## KitchenAbvGr   KitchenAbvGr  0.018216102
## MasVnrArea       MasVnrArea  0.014884062
## Electrical       Electrical  0.013951298
## RoofStyle         RoofStyle  0.013066655
## BedroomAbvGr   BedroomAbvGr  0.011976756
## BsmtFinSF2       BsmtFinSF2  0.011382589
## MSSubClass       MSSubClass  0.011022616
## GarageCond       GarageCond  0.010029772
## BsmtFinType2   BsmtFinType2  0.009231118
## Condition2       Condition2  0.006200024
## GarageQual       GarageQual  0.005903172
## MiscVal             MiscVal  0.005747103
## LowQualFinSF   LowQualFinSF  0.005721744
## LotShape           LotShape  0.003636011
## LandSlope         LandSlope  0.003094387
## BsmtHalfBath   BsmtHalfBath  0.001680458
## Street               Street  0.000000000
## Utilities         Utilities  0.000000000
## MasVnrType       MasVnrType  0.000000000
## Heating             Heating  0.000000000
## X3SsnPorch       X3SsnPorch  0.000000000
## PoolArea           PoolArea  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: 0.0931
```

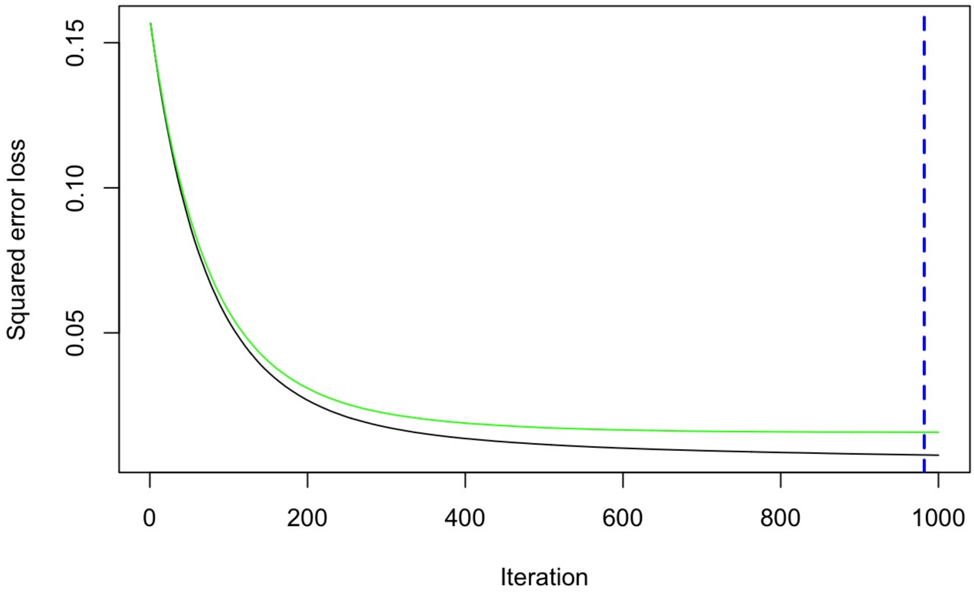
```r
# Convert predictions back to original price scale
preds <- expm1(log_preds)
actuals <- expm1(log_actuals)

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

```
##  Dollar-scale RMSE: 18809.94
```

```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]])
  }
}

# ------------------ 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)

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): 0.1303
```

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

```
##  Approximate Percent Error: 13.91 %
```

```
# 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): 35876.65
```

```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.
```