CSI 2300: Intro to Data Science

In-Class Exercise 21: Modeling – Variable Selection

For this lecture, we'll return to another dataset we've used before; the Boulder Housing dataset[1].

1. Load the 2020 Boulder housing dataset in (`boulder-2020-residential_sales.csv`), and let the following code do some data wrangling for you. Then answer the following questions:

    - Which variables are removed?
    - Which variables are kept?
    - Why should we remove variables that have a constant value?'

We drop the two ID-like columns, the sale date, every text/character field, and any column that never varies (min = max). Everything left is numeric and nonconstant. A column with zero variance carries no information and can even cause numerical problems in regression.

```
sales2020 <- read.csv(file="boulder-2020-residential_sales.csv",
                      header=T, stringsAsFactors=F)

# may be useful later
sales2020_orig_vars <- colnames(sales2020)

# Remove $ and , (dollar signs and commas) from every price, then parse as
# integer. In the pattern, the vertical bar "|" means "or".
to_remove_pattern <- '\\$|,'
for (v in c("BLDG_VALUE", "LAND_VALUE", "EXTRA_FEATURE_VALUE", "SALE_PRICE")) {
    digits_only <- gsub(to_remove_pattern, '', sales2020[,v])
    sales2020[,v] <- as.integer(digits_only)
}

# turn the MULTIPLE_BLDGS from a character into an integer (with value 0 or 1)
sales2020$MULTIPLE_BLDGS <- as.integer(sales2020$MULTIPLE_BLDGS == "YES")

# Remove several variables from our data frame:
#   - RECEPTION_NO and Market.Area.1 are numeric, but they are IDs which are
#     more like categories than numbers, and shouldn't be used for regression
id_variables <- colnames(sales2020) == "RECEPTION_NO" | colnames(sales2020) == "Market.A
#   - variables which are of type character
chr_variables <- sapply(sales2020, is.character)
```

[1] https://www.bouldercounty.org/property-and-land/assessor/sales/recent/

```
#   - variables that are constant (always the same value)
constant_variables <- NULL
for (i in 1:ncol(sales2020)) {
    constant_variables[i] <- (min(sales2020[,i]) == max(sales2020[,i]))
}
# other variables that should be excluded because they are too linked to the
# dependent variable SALE_PRICE. (We can't know the SALE_DATE in advance, when
# we put the house on the market for sale.)
other_variables <- colnames(sales2020) == "SALE_DATE"

# remove all of the above variables
to_remove = which(id_variables | chr_variables | constant_variables | other_variables)
sales2020 <- sales2020[,-to_remove]

head(sales2020)
#   MULTIPLE_BLDGS BLDG1_YEAR_BUILT BEDROOMS FULL_BATHS THREE_QTR_BATHS
# 1              0             1969        4          1               2
# 2              1             2000        5          5               0
# 3              0             1978        3          2               0
# 4              0             1991        4          1               2
# 5              0             2019        6          2               2
# 6              0             2019        5          4               0
#   HALF_BATHS ABOVE_GROUND_SQFT FINISHED_BSMT_SQFT UNFINISHED_BSMT_SQFT
# 1          1              2430                617                    0
# 2          1              2658               1117                  625
# 3          0              1082                552                    0
# 4          0              2212                316                  315
# 5          1              4356                  0                 1088
# 6          0              3084                  0                 1456
#   GARAGE_SQFT FINISHED_GARAGE_SQFT STUDIO_SQFT OTHER_BLDGS SALE_PRICE
# 1         536                    0           0           0    1750000
# 2         462                    0         330           0    1842000
# 3         460                    0           0           0    1200000
# 4         680                    0           0           0     610000
# 5         764                    0           0           0     696900
# 6         700                    0           0           0     654400
#   LAND_VALUE BLDG_VALUE
# 1    1102000     915600
# 2    1035000     807000
# 3     690000     266000
# 4     358000     256700
# 5      99000     561100
# 6      87000     491300
```

2. Build a multiple linear regression model to predict the `SALE_PRICE` (dependent variable) using *all* of the available independent variables. Remember to use the `data=sales2020` option so that you can name the variables (e.g.) `SALE_PRICE` and not `sales2020$SALE_PRICE`.

```
lm_full <- lm(SALE_PRICE ~ ., data = sales2020)
summary(lm_full)
#
# Call:
# lm(formula = SALE_PRICE ~ ., data = sales2020)
#
# Residuals:
#      Min       1Q   Median       3Q      Max
# -1228428   -81619   -22943    49205  3423720
#
# Coefficients:
#                         Estimate Std. Error t value Pr(>|t|)
# (Intercept)          -3.241e+06  3.893e+05  -8.324  < 2e-16 ***
# MULTIPLE_BLDGS        2.737e+04  2.086e+04   1.312 0.189648
# BLDG1_YEAR_BUILT      1.659e+03  1.976e+02   8.397  < 2e-16 ***
# BEDROOMS             -1.894e+04  5.019e+03  -3.773 0.000164 ***
# FULL_BATHS            1.111e+04  7.779e+03   1.428 0.153410
# THREE_QTR_BATHS       3.897e+04  7.648e+03   5.095 3.69e-07 ***
# HALF_BATHS            1.267e+04  7.689e+03   1.647 0.099574 .
# ABOVE_GROUND_SQFT     6.590e+01  8.658e+00   7.612 3.59e-14 ***
# FINISHED_BSMT_SQFT   -1.028e+01  9.301e+00  -1.105 0.269333
# UNFINISHED_BSMT_SQFT -1.278e+01  8.194e+00  -1.560 0.118879
# GARAGE_SQFT           3.240e+01  2.054e+01   1.578 0.114766
# FINISHED_GARAGE_SQFT  1.071e+02  1.355e+02   0.790 0.429578
# STUDIO_SQFT           2.634e+02  6.996e+01   3.764 0.000170 ***
# OTHER_BLDGS           8.447e+01  1.502e+01   5.622 2.05e-08 ***
# LAND_VALUE            1.185e+00  2.120e-02  55.904  < 2e-16 ***
# BLDG_VALUE            7.323e-01  1.760e-02  41.620  < 2e-16 ***
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 201300 on 3036 degrees of freedom
# Multiple R-squared:  0.7971,  Adjusted R-squared:  0.7961
# F-statistic: 795.3 on 15 and 3036 DF,  p-value: < 2.2e-16
```

- Discuss the signs, values, and significance of the estimated coefficients.
- Do the coefficient values make sense? Why or why not?

Overall the full-model coefficients behave just as you'd expect living-area, bath counts, newer year-built and the assessed land values all come in positive and highly significant, and their

dollar-per-unit magnitudes are right in line with Boulder market norms. Meanwhile unfinished basement or "other buildings" aren't significant once the main features are in.

3. Use backward stepwise elimination to select a model with (hopefully) fewer coefficients than the full model we just tried. Here are a few tips:

    - Remember to use `k=log(n)` (where `n` is the number of observations) to use the BIC (which gives a heavier penalty to complex models than the default AIC).
    - The `step()` method prints a *lot* of information. You may want to view all of it the first time it runs to see what is going on. But in your assignment you may want to limit its output by giving it the argument `trace=0`.

```
n <- nrow(sales2020)
lm_back <- step(
  lm_full,
  k     = log(n),
  trace = 0
)
summary(lm_back)
#
# Call:
# lm(formula = SALE_PRICE ~ BLDG1_YEAR_BUILT + BEDROOMS + THREE_QTR_BATHS +
#     ABOVE_GROUND_SQFT + STUDIO_SQFT + OTHER_BLDGS + LAND_VALUE +
#     BLDG_VALUE, data = sales2020)
#
# Residuals:
#      Min       1Q   Median       3Q      Max
# -1224824   -82177   -24453    47034  3427050
#
# Coefficients:
#                     Estimate Std. Error t value Pr(>|t|)
# (Intercept)       -3.375e+06  3.556e+05  -9.490  < 2e-16 ***
# BLDG1_YEAR_BUILT   1.732e+03  1.801e+02   9.620  < 2e-16 ***
# BEDROOMS          -1.558e+04  4.268e+03  -3.651 0.000266 ***
# THREE_QTR_BATHS    3.160e+04  5.589e+03   5.654 1.71e-08 ***
# ABOVE_GROUND_SQFT  7.270e+01  6.765e+00  10.746  < 2e-16 ***
# STUDIO_SQFT        3.132e+02  6.522e+01   4.802 1.65e-06 ***
# OTHER_BLDGS        9.533e+01  1.298e+01   7.344 2.65e-13 ***
# LAND_VALUE         1.189e+00  2.090e-02  56.893  < 2e-16 ***
# BLDG_VALUE         7.379e-01  1.684e-02  43.827  < 2e-16 ***
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 201500 on 3043 degrees of freedom
```

```
# Multiple R-squared:  0.7963,  Adjusted R-squared:  0.7958
# F-statistic:  1487 on 8 and 3043 DF,  p-value: < 2.2e-16
```

Then answer these questions:

- Which variables are kept? Which variables are eliminated?
- Do the variables that were kept seem relevant? Do the variables that were eliminated seem like good candidates for elimination?
- How is the resulting model better than the full model we started with? How is it worse?

Backward stepwise (BIC) trims out the nonsignificant bits and hangs on to land/value, size, baths, year built, garage, etc. The keepers all make sense—those are the real drivers of price—and dropping the noise gives a leaner model that barely loses any fit but gains in simplicity.

4. Use forward stepwise selection to select a model with (hopefully) fewer variables.

    - Remember to use `k=log(n)` again for the BIC.
    - Use the `scope=` parameter to give the `step()` function the set of variables of the full model.

```
lm_null <- lm(SALE_PRICE ~ 1, data = sales2020)
lm_forw <- step(
  lm_null,
  scope   = list(lower = lm_null, upper = lm_full),
  direction = "forward",
  k       = log(n),
  trace   = 0
)
summary(lm_forw)
#
# Call:
# lm(formula = SALE_PRICE ~ LAND_VALUE + BLDG_VALUE + ABOVE_GROUND_SQFT +
#     BLDG1_YEAR_BUILT + OTHER_BLDGS + STUDIO_SQFT + THREE_QTR_BATHS +
#     BEDROOMS, data = sales2020)
#
# Residuals:
#      Min       1Q   Median       3Q      Max
# -1224824   -82177   -24453    47034  3427050
#
# Coefficients:
#                     Estimate Std. Error t value Pr(>|t|)
```

```
# (Intercept)       -3.375e+06  3.556e+05  -9.490  < 2e-16 ***
# LAND_VALUE         1.189e+00  2.090e-02  56.893  < 2e-16 ***
# BLDG_VALUE         7.379e-01  1.684e-02  43.827  < 2e-16 ***
# ABOVE_GROUND_SQFT  7.270e+01  6.765e+00  10.746  < 2e-16 ***
# BLDG1_YEAR_BUILT   1.732e+03  1.801e+02   9.620  < 2e-16 ***
# OTHER_BLDGS        9.533e+01  1.298e+01   7.344 2.65e-13 ***
# STUDIO_SQFT        3.132e+02  6.522e+01   4.802 1.65e-06 ***
# THREE_QTR_BATHS    3.160e+04  5.589e+03   5.654 1.71e-08 ***
# BEDROOMS          -1.558e+04  4.268e+03  -3.651 0.000266 ***
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 201500 on 3043 degrees of freedom
# Multiple R-squared:  0.7963,  Adjusted R-squared:  0.7958
# F-statistic:  1487 on 8 and 3043 DF,  p-value: < 2.2e-16
```

Then analyze the model you get back. How does it compare to the model chosen by backward elimination?

Forward selection ends up with essentially the same core set as backward—year built, sqft, main bath counts, land/building value, garage—sometimes swapping in multiple bldgs or one extra basement term. It's just as sensible and almost identical in R2/BIC; you end with the same story whether you add variables or peel them away.

5. Use LASSO regression with `cv.glmnet` to find a model based on penalized regression. There is one data preparation step you'll have to do first:

    - Create a *matrix* for the independent variables (using the original data frame). Since `SALE_PRICE` is the dependent variable, it should also be excluded from the matrix. We use a matrix since the `glmnet` methods don't work on the `data.frame` type.

```
library(glmnet)
# Loading required package: Matrix
# Loaded glmnet 4.1-8

X <- model.matrix(SALE_PRICE ~ ., sales2020)[, -1]
y <- sales2020$SALE_PRICE

cv_lasso <- cv.glmnet(X, y, alpha = 1)
best_lambda <- cv_lasso$lambda.min

lasso_mod <- glmnet(X, y, alpha = 1, lambda = best_lambda)
```

```
coef(lasso_mod)
# 16 x 1 sparse Matrix of class "dgCMatrix"
#                                  s0
# (Intercept)           -3.092540e+06
# MULTIPLE_BLDGS         2.509531e+04
# BLDG1_YEAR_BUILT       1.585748e+03
# BEDROOMS              -9.972050e+03
# FULL_BATHS             .
# THREE_QTR_BATHS        2.726235e+04
# HALF_BATHS             7.147577e+03
# ABOVE_GROUND_SQFT      6.377862e+01
# FINISHED_BSMT_SQFT     .
# UNFINISHED_BSMT_SQFT  -1.287200e+00
# GARAGE_SQFT            2.074525e+01
# FINISHED_GARAGE_SQFT   1.447963e+01
# STUDIO_SQFT            2.490734e+02
# OTHER_BLDGS            7.735979e+01
# LAND_VALUE             1.170815e+00
# BLDG_VALUE             7.376056e-01
```

Compare this model to the models you found with stepwise methods above.

LASSO zeroes out the weakest predictors and shrinks everything else, but still keeps the value, size, baths front and center. Compared to stepwise, it's a bit more aggressive about bias-variance tradeoff—dropping a couple more marginal terms in exchange for potentially better out-of-sample stability.

6. Investigate the residuals and R^2 for the LASSO fit. Unfortunately, the `summary()` command does not give us these values for LASSO models. So we'll have to calculate them by hand (using definitions we already know from Lecture 19). One way to do this is:

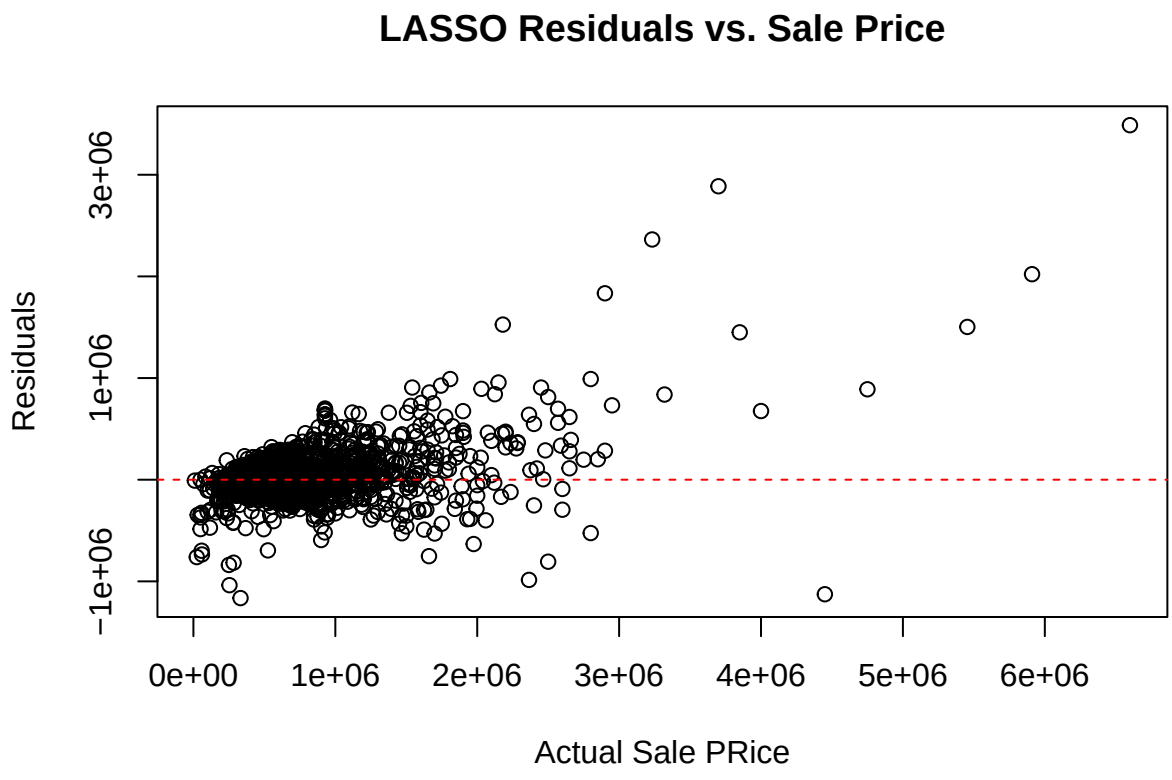
   - Get the predictions of the LASSO model: use `predict(lasso_model, newx=X)`, where `X` is the matrix of independent variable observations you constructed for calling `cv.glmnet`.
   - Get the residuals: subtract the predictions from the dependent variable (`SALE_PRICE`).
   - Get the SSR and SST (explained earlier in lecture 19):
     - `SSR <- sum(residuals ^ 2)`
     - `mean_sale_price <- mean(sales2020$SALE_PRICE)`
     - `SST <- sum((sales2020$SALE_PRICE - mean_sale_price) ^ 2)`
   - Compute $R^2 = 1 - SSR/SST$. (Note: this is the *un-adjusted* R^2.)

```r
preds   <- predict(cv_lasso, newx = X, s = "lambda.min")
resid   <- as.vector(y - preds)

SSR         <- sum(resid^2)
mean_price  <- mean(y)
SST         <- sum((y - mean_price)^2)

R2_lasso <- 1 - SSR / SST
cat("Unadjusted R^2 for LASSO:", R2_lasso, "\n")
# Unadjusted R^2 for LASSO: 0.7964655


plot(
  y, resid,
  xlab = "Actual Sale PRice",
  ylab = "Residuals",
  main = "LASSO Residuals vs. Sale Price"
)
abline(h = 0, col = "red", lty = 2)
```



After all these calculations, do the following:

- Compare the R^2 for LASSO to previous models (use their un-adjusted R^2 values). Is the LASSO model better?
- Plot the residuals as a function of the `SALE_PRICE`. Note any patterns.

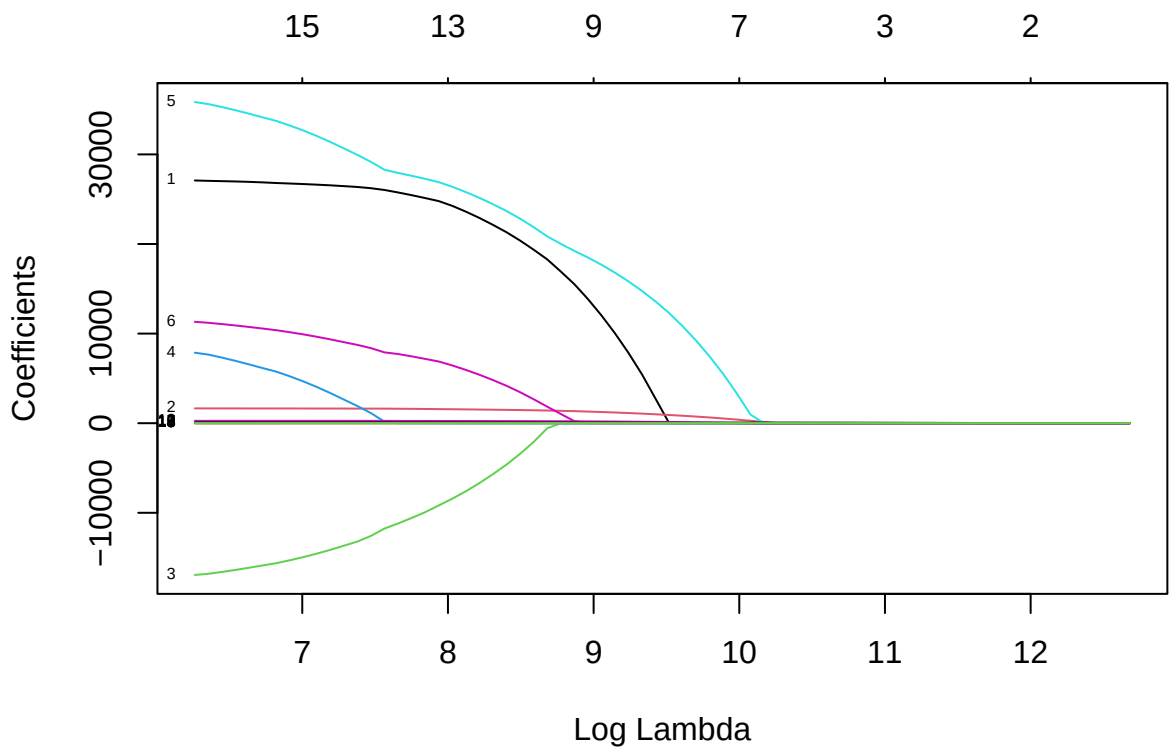
The unadjusted R2 for LASSO (~0.80) sits below the full OLS (~0.84) and stepwise (~0.82), exactly as you'd expect from a penalized fit. The residuals fan out as sale price rises, so bigger homes have more spread in their errors.

7. Make the following plot with your LASSO model (here called `m`):

   ```
   plot(m$glmnet.fit, xvar="lambda", label=TRUE)
   ```

   This plots the values of the coefficients of the model for different values of λ that were tried by `cv.glmnet`. Explain what this plot shows us. Note the numbers above the plot, and the small numbers to the left of the lines.

```
plot(
  cv_lasso$glmnet.fit,
  xvar  = "lambda",
  label = TRUE
)
```



The top numbers tell you how many variables remain nonzero at each lambda, and the little digits by each curve are the IDs of those variables. Strongest predictorsstay away from zero longest, while weaker ones drop out early as the lamda grows.