

# Variable Selection Methods: Regularization with Ridge and LASSO

August 17, 2024

## Table of contents

Recap . . . . .	1
Motivation . . . . .	2
Objectives . . . . .	3
Ordinary Least Squares . . . . .	3
Ridge Regression . . . . .	4
Least Absolute Shrinkage and Selection Operator (LASSO) . . . . .	4
Some Feature PreProcessing Concerns . . . . .	5
Fitting a Model Using Ridge Regression or the LASSO . . . . .	5
Ummm... Great – So What Do I Use and When Do I Use It? . . . . .	6
Implementing Ridge Regression and the LASSO . . . . .	6
Implementing Ridge Regression . . . . .	7
Implementing the LASSO . . . . .	12
Summary . . . . .	17

## Recap

With Cross-Validation, we've supercharged our modeling powers. We've made our model performance estimates much more reliable, which gives us greater confidence in understanding the predictive power (or lack thereof) in our models. We've become much more responsible modelers, in this sense. We'll see again soon that cross-validation can actually do much more for us than just provide more reliable performance estimates than the validation-set approach did. For now though, we shift focus slightly to the question – *how do we know which predictors should be included in our model?*

## Motivation

When we first introduced multiple linear regression, we took an approach in which we built a “large” model – a model that included most/all of our available predictors. Once we had that model, we used a procedure called *backward elimination* to eliminate terms from our model if they were associated with  $p$ -values above the 5% significance threshold. We could have taken the opposite approach, starting with an empty model, adding predictors/terms one at a time – a process called *forward selection*.

With both of these methods, we’re allowing our model to be quite greedy. It may be helpful to think of these processes as allowing our model to go “shopping” for predictors – in the *backward selection* paradigm, the model begins by adding every item in the store to its shopping cart, and then carefully places the ones it doesn’t want back on the shelves – in the *forward selection* paradigm, the model begins with an empty cart and adds its favorite items one-by-one into its shopping cart. This seems quite reasonable at first, but we’re engaging in quite risky behavior.

- From a purely statistical standpoint, we’re evaluating lots of  $t$ -tests in determining whether model terms are statistically significant or not. The likelihood of making at least one Type I Error (saying that a model term is statistically significant, when it is not so in the population) becomes very large in these processes.
- From a model-fit perspective, the more predictor variables a model has, the more flexible it is – the more flexible a model is, the better it will fit the *training data* and the more likely it is to become overfit.

By allowing our model to “shop” freely for its predictors, we are enticing our model to overfit the training data – giving our model a “budget” to spend on its shopping trip would be a really nice way to lower the likelihood that our model “buys” too many predictors and overfits the training data.

### **i** Optimization Procedures for Model Fitting

All of the models we’ve fit so far in our course use *Ordinary Least Squares* as the underlying fitting procedure – we are moving to new waters now. Ridge Regression and LASSO models belong to a class of model called *Generalized Linear Models*, although their fitting procedures under the hood will be different, the consistency of the `{tidymodels}` framework allows us to make this change quite simply.

### **i** Regularization

The process of applying an additional constraint such as a *budget* one example of model constraints commonly referred to as *regularization*. Regularization techniques are utilized with lots of model classes to help prevent those models from overfitting our training data.

## Objectives

After working through this notebook you should be able to:

- Articulate the *backward selection* approach and how it relates to the multiple linear regression models we've constructed and analyzed so far in our course.
- Discuss *forward selection* as an alternative to the backward selection process.
- Discuss and implement *Ridge Regression* and the *LASSO* as alternatives to Ordinary Least Squares within the `{tidymodels}` framework.
- Discuss the benefits and drawbacks for forward- and backward-selection, Ridge Regression, and the LASSO relative to one another.

---

## Ordinary Least Squares

The process of fitting a model using ordinary least squares is an *optimization* problem. Simply put, to fit a model of the form

$$\mathbb{E}[y] = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k$$

we minimize the residual sum of squares:

$$\text{Minimize: } \sum_{\text{training data}} (y_{\text{observed}} - y_{\text{predicted}})^2$$

Which is equivalent to

$$\text{Minimize: } \sum_{\text{training data}} \left( y_{\text{observed}} - \left( \beta_0 + \sum_{i=1}^k \beta_i x_i \right) \right)^2$$

This process of choosing values for  $\beta_0, \beta_1, \beta_2, \dots, \beta_k$  is called *Ordinary Least Squares* (OLS), and it's what we've been using (or, rather R has been using) all semester long to fit our models.

In OLS, the optimization procedure can freely choose those  $\beta$  values, without any constraint (OLS is shopping without a budget). Ridge Regression and the LASSO introduce extra constraints on this optimization problem.

## Ridge Regression

The process of fitting a model using Ridge Regression is similar to that of using OLS, except that a constraint on the chosen  $\beta$  coefficients is imposed. That is, using Ridge Regression to fit a model of the form

$$\mathbb{E}[y] = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k$$

Solves the following optimization problem:

$$\begin{aligned} \text{Minimize:} & \sum_{\text{training data}} \left( y_{\text{observed}} - \left( \beta_0 + \sum_{i=1}^k \beta_i x_i \right) \right)^2 \\ \text{Subject To:} & \sum_{i=1}^k |\beta_i| \leq C \end{aligned}$$

Where  $C$  can be thought of as a total coefficient budget. That is, it becomes very expensive for the model to assign lots of non-zero coefficients.

## Least Absolute Shrinkage and Selection Operator (LASSO)

The difference between Ridge Regression and the LASSO is in how “spent budget” is calculated. Instead of summing the absolute values of the  $\beta$  coefficients, we’ll sum their squares. That is, using LASSO to fit a model of the form

$$\mathbb{E}[y] = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k$$

Solves the following optimization problem:

$$\begin{aligned} \text{Minimize:} & \sum_{\text{training data}} \left( y_{\text{observed}} - \left( \beta_0 + \sum_{i=1}^k \beta_i x_i \right) \right)^2 \\ \text{Subject To:} & \sum_{i=1}^k \beta_i^2 \leq C \end{aligned}$$

Where  $C$  can again be thought of as a total coefficient budget. Because it squares the  $\beta$  coefficients in the calculation of “budget used”, the LASSO penalizes large coefficients more than Ridge Regression does. Furthermore, due to the mathematics behind these algorithms, models fit using the LASSO result in small coefficients being sent to 0. That is, the LASSO can be used as a *variable selection procedure*. Models fit with Ridge Regression often leave several predictors having very small coefficients so, while they don’t have much influence in the overall model predictions, they do still remain in the model.

## Some Feature PreProcessing Concerns

If we are going to utilize Ridge Regression or the LASSO, we need to ensure that all of our predictors are on a standardized scale. Otherwise, predictors/features whose observed values are large are superficially *cheaper* for the model to use (because we can attach small coefficients to them) than features whose observed values are already small. Similarly, features whose observed values are very small are artificially made *more expensive* for the model to use because they demand larger coefficients in order to have influence over model predictions. There are two very common scaling methods:

- Min/Max scaling projects a variable onto the interval  $[0, 1]$ , where the minimum observed value is sent to 0 and the maximum observed value is sent to 1.

- Min/Max scaling for the variable  $x$  is done using the formula:  $\frac{x - \min(x)}{\max(x) - \min(x)}$ .

- We can add `step_range()` to a `recipe()` to min/max scale a feature.

- Standardization converts a variables raw measurements into standard deviations ( $z$ -scores), where the mean of the transformed variable is 0 and the standard deviation of the transformed variable is 1.

- Standardized scaling for the variable  $x$  is done using the formula:  $\frac{x - \text{mean}(x)}{\text{sd}(x)}$

- We can add `step_normalize()` to a `recipe()` to standardize the values of a feature.

## Fitting a Model Using Ridge Regression or the LASSO

The `tidymodels` framework provides us with a standardized structure for defining and fitting models. Everything we've learned about the syntax and workflow for fitting a linear regression model using OLS can be used to fit these Generalized Linear Models. We'll just need to `set_engine()` to something other than "lm" – specifically, an engine which can fit models using the constraints defined in the Ridge Regression and LASSO sections above. We'll use "glmnet", which requires two additional arguments `mixture` and `penalty`.

- The `mixture` argument is a number between 0 and 1. Setting `mixture = 1` results in a LASSO model, while setting `mixture = 0` results in a Ridge Regression model. Values in between 0 and 1 result in a *mixed* LASSO/Ridge approach, where the penalty is partially determined by the Ridge Regression constraint and partially by the LASSO constraint.
- The `penalty` argument corresponds to the amount of *regularization* being applied – that is, the `penalty` is related to the *budget* parameter we've described earlier.

## Ummm... Great – So What Do I Use and When Do I Use It?

There is some really good news coming – just bear with me!

The choice between Ridge Regression and LASSO depends on your goals. If you are looking for a tool to help with *variable selection*, then the LASSO is your choice, since it will result in less valuable predictors being assigned coefficients of exactly 0. If you are building a predictive model, you might try both and see which one performs better for your specific use-case.

How do I choose a **penalty**? As I understand it, there's no great science to choosing a penalty. People will typically try values and see what the best ones are for their particular use-case.

So basically I've told you about these two new classes of model that help prevent overfitting. They each require a **penalty** parameter and I've given you no real guidance on how to choose it. For now, we'll simply choose a **penalty** parameter and mention what we could build several models with different **penalty** values to try and find one that performs best. I'll give you a much better method when we talk about hyperparameter tuning.

## Implementing Ridge Regression and the LASSO

There are a few additional concerns with fitting Ridge Regression and LASSO models that we haven't needed to deal with prior.

- The `glmnet` engine requires that no missing values are included in any fold.
  - We can *omit* rows with missing data. (drawback: we are throwing away observations and we may be drastically reducing the size of our available data)
  - We can *omit* features with missing data. (drawback: we are sacrificing potentially valuable predictors by omitting them from our model)
  - We can *impute* missing values using a `step_impute_*`() feature engineering step in our recipe. (drawback: we are making our best guess at what the missing value should be, but we are introducing additional uncertainty to our model)
- Because the **penalty** parameter imposes a *budget* on coefficients, all of our predictors must be on the same scale – otherwise, some predictors are artificially *cheaper* or *more expensive* than others to include in our model.
  - We can use `step_range()` or `step_normalize()` on `all_numerical_predictors()` to achieve this.

We'll filter out the observations with an unknown response (`Sale_Price`), and we'll use `step_impute_knn()` to use a *k*-nearest neighbors imputation scheme for any missing values in our predictor columns. Finally, we'll use `step_normalize()` to center and scale our numerical predictors. Then we'll proceed as usual.

## Implementing Ridge Regression

We'll build and assess a Ridge Regression model first.

```
set.seed(123)
ames_known_price <- ames %>%
  filter(!is.na(Sale_Price))
ames_split <- initial_split(ames_known_price, prop = 0.9)
ames_train <- training(ames_split)
ames_test <- testing(ames_split)

ames_folds <- vfold_cv(ames_train, v = 5)

ridge_reg_spec <- linear_reg(mixture = 0, penalty = 0.1) %>%
  set_engine("glmnet")

reg_rec <- recipe(Sale_Price ~ ., data = ames_train) %>%
  step_normalize(all_numeric_predictors()) %>%
  step_impute_knn(all_predictors()) %>%
  step_normalize(all_numeric_predictors()) %>%
  step_other(all_nominal_predictors()) %>%
  step_dummy(all_nominal_predictors())

ridge_reg_wf <- workflow() %>%
  add_model(ridge_reg_spec) %>%
  add_recipe(reg_rec)

ridge_reg_cv <- ridge_reg_wf %>%
  fit_resamples(ames_folds)

ridge_reg_cv %>%
  collect_metrics() %>%
  kable() %>%
  kable_styling(bootstrap_options = c("striped", "hover"))
```

.metric	.estimator	mean	n	std_err	.config
rmse	standard	3.163600e+04	5	3258.9443024	Preprocessor1_Model1
rsq	standard	8.477678e-01	5	0.0205856	Preprocessor1_Model1

```

ridge_reg_cv %>%
  collect_metrics(summarize = FALSE) %>%
  kable() %>%
  kable_styling(bootstrap_options = c("striped", "hover"))

```

id	.metric	.estimator	.estimate	.config
Fold1	rmse	standard	4.023294e+04	Preprocessor1_Model1
Fold1	rsq	standard	8.121570e-01	Preprocessor1_Model1
Fold2	rmse	standard	3.227439e+04	Preprocessor1_Model1
Fold2	rsq	standard	8.499513e-01	Preprocessor1_Model1
Fold3	rmse	standard	2.373823e+04	Preprocessor1_Model1
Fold3	rsq	standard	8.965585e-01	Preprocessor1_Model1
Fold4	rmse	standard	3.708400e+04	Preprocessor1_Model1
Fold4	rsq	standard	7.915890e-01	Preprocessor1_Model1
Fold5	rmse	standard	2.485042e+04	Preprocessor1_Model1
Fold5	rsq	standard	8.885830e-01	Preprocessor1_Model1

```

ridge_reg_fit <- ridge_reg_wf %>%
  fit(ames_train)

ridge_reg_fit %>%
  tidy() %>%
  kable() %>%
  kable_styling(bootstrap_options = c("striped", "hover"))

```

term	estimate	penalty
Intercept	191818.9719	0.1
Lot_Frontage	635.0426	0.1
Lot_Area	2632.2748	0.1
Year_Built	3643.3324	0.1
Year_Remod_Add	4676.9398	0.1
Mas_Vnr_Area	4539.0144	0.1
BsmtFin_SF_1	-428.1522	0.1
BsmtFin_SF_2	792.9066	0.1
Bsmt_Unf_SF	-2277.7734	0.1
Total_Bsmt_SF	5900.9613	0.1
First_Flr_SF	7349.9085	0.1
Second_Flr_SF	9836.1873	0.1



Low_Qual_Fin_SF	-524.0123	0.1
Gr_Liv_Area	13599.1103	0.1
Bsmt_Full_Bath	2770.9347	0.1
Bsmt_Half_Bath	-658.8395	0.1
Full_Bath	4052.4543	0.1
Half_Bath	2013.8465	0.1
Bedroom_AbvGr	-1283.6240	0.1
Kitchen_AbvGr	-2450.6191	0.1
TotRms_AbvGrd	2934.1361	0.1
Fireplaces	4970.9791	0.1
Garage_Cars	5526.8513	0.1
Garage_Area	2106.3876	0.1
Wood_Deck_SF	1412.6677	0.1
Open_Porch_SF	-525.2878	0.1
Enclosed_Porch	1394.0973	0.1
Three_season_porch	495.8018	0.1
Screen_Porch	3635.3907	0.1
Pool_Area	-1409.9219	0.1
Misc_Val	-5134.1938	0.1
Mo_Sold	-454.5277	0.1
Year_Sold	-1282.1268	0.1
Longitude	988.5822	0.1
Latitude	4725.5739	0.1
MS_SubClass_One_Story_1945_and_Older	-4416.9538	0.1
MS_SubClass_One_Story_1946_and_Newer_All_Styles	3304.0249	0.1
MS_SubClass_One_Story_PUD_1946_and_Newer	-3143.7530	0.1
MS_SubClass_Two_Story_1946_and_Newer	324.7893	0.1
MS_SubClass_other	1715.7145	0.1
MS_Zoning_Residential_Medium_Density	-5868.1882	0.1
MS_Zoning_other	-8029.4040	0.1
Street_other	-23696.6799	0.1
Alley_other	-104.5784	0.1
Lot_Shape_Slightly_Irregular	2245.2842	0.1
Lot_Shape_other	1938.9597	0.1
Land_Contour_other	-408.0211	0.1
Utilities_other	-13984.6732	0.1
Lot_Config_CulDSac	10373.2126	0.1
Lot_Config_Inside	376.8869	0.1
Lot_Config_other	-5855.1299	0.1

Land_Slope_other	5012.1091	0.1
Neighborhood_Edwards	-9390.3854	0.1
Neighborhood_Gilbert	-16966.9174	0.1
Neighborhood_North_Ames	-5558.3609	0.1
Neighborhood_Northridge_Heights	18265.8881	0.1
Neighborhood_Old_Town	-7357.2823	0.1
Neighborhood_Somerset	9460.1468	0.1
Neighborhood_other	4718.5160	0.1
Condition_1_Norm	10833.0680	0.1
Condition_1_other	2692.3090	0.1
Condition_2_other	-388.4195	0.1
Bldg_Type_TwnhsE	-12433.2155	0.1
Bldg_Type_other	-12772.5323	0.1
House_Style_One_Story	3970.5877	0.1
House_Style_Two_Story	-3677.5124	0.1
House_Style_other	-1127.9941	0.1
Overall_Qual_Average	-1706.9615	0.1
Overall_Qual_Below_Average	-2948.3523	0.1
Overall_Qual_Good	1648.5617	0.1
Overall_Qual_Very_Good	15406.4636	0.1
Overall_Qual_other	29505.5762	0.1
Overall_Cond_Average	-2439.3189	0.1
Overall_Cond_Good	4982.5093	0.1
Overall_Cond_other	-3768.5434	0.1
Roof_Style_Hip	4932.1920	0.1
Roof_Style_other	-4617.6292	0.1
Roof_Matl_other	1693.2505	0.1
Exterior_1st_MetalSd	2168.7468	0.1
Exterior_1st_Plywood	795.5784	0.1
Exterior_1st_VinylSd	851.9245	0.1
Exterior_1st_Wd.Sdng	-186.6109	0.1
Exterior_1st_other	6571.8953	0.1
Exterior_2nd_MetalSd	1596.0933	0.1
Exterior_2nd_Plywood	-3631.0092	0.1
Exterior_2nd_VinylSd	949.2826	0.1
Exterior_2nd_Wd.Sdng	3755.8799	0.1
Exterior_2nd_other	-1258.0924	0.1
Mas_Vnr_Type_None	6482.2932	0.1
Mas_Vnr_Type_Stone	2906.4064	0.1

Mas_Vnr_Type_other	-8363.1220	0.1
Exter_Qual_Typical	-6484.2417	0.1
Exter_Qual_other	15366.6626	0.1
Exter_Cond_Typical	-857.9075	0.1
Exter_Cond_other	-14473.8030	0.1
Foundation_CBlock	-1773.5303	0.1
Foundation_PConc	4484.0142	0.1
Foundation_other	320.5474	0.1
Bsmt_Qual_Good	-15251.3685	0.1
Bsmt_Qual_Typical	-14338.6875	0.1
Bsmt_Qual_other	-14341.2896	0.1
Bsmt_Cond_other	-2799.6797	0.1
Bsmt_Exposure_Gd	13803.2479	0.1
Bsmt_Exposure_Mn	-5658.9405	0.1
Bsmt_Exposure_No	-8614.6482	0.1
Bsmt_Exposure_other	-6281.7374	0.1
BsmtFin_Type_1_BLQ	-511.8263	0.1
BsmtFin_Type_1_GLQ	7199.5285	0.1
BsmtFin_Type_1_LwQ	-4358.6347	0.1
BsmtFin_Type_1_Rec	-849.0571	0.1
BsmtFin_Type_1_Unf	-1914.3319	0.1
BsmtFin_Type_1_other	-2926.1817	0.1
BsmtFin_Type_2_other	-2019.5929	0.1
Heating_other	-1729.5089	0.1
Heating_QC_Good	-2492.5041	0.1
Heating_QC_Typical	-5346.1460	0.1
Heating_QC_other	-10686.6811	0.1
Central_Air_Y	4072.2257	0.1
Electrical_SBrkr	346.4420	0.1
Electrical_other	301.2078	0.1
Kitchen_Qual_Good	-13742.2922	0.1
Kitchen_Qual_Typical	-16186.8202	0.1
Kitchen_Qual_other	-18648.4382	0.1
Functional_other	-15944.3071	0.1
Fireplace_Qu_No_Fireplace	542.6519	0.1
Fireplace_Qu_Typical	-1928.1394	0.1
Fireplace_Qu_other	646.1564	0.1
Garage_Type_BuiltIn	-941.3937	0.1
Garage_Type_Detchd	-1661.7981	0.1
Garage_Type_No_Garage	617.5876	0.1

Garage_Type_other	-11561.5629	0.1
Garage_Finish_No_Garage	612.2936	0.1
Garage_Finish_RFn	-3891.0061	0.1
Garage_Finish_Unf	-1754.4046	0.1
Garage_Qual_Typical	-1528.9216	0.1
Garage_Qual_other	2247.5272	0.1
Garage_Cond_Typical	2349.5782	0.1
Garage_Cond_other	-6338.7852	0.1
Paved_Drive_Paved	4236.6662	0.1
Paved_Drive_other	5735.1255	0.1
Pool_QC_other	10854.0122	0.1
Fence_No_Fence	-2304.5540	0.1
Fence_other	-1200.1668	0.1
Misc_Feature_other	6997.8461	0.1
Sale_Type_WD	-4789.3508	0.1
Sale_Type_other	-6853.4085	0.1
Sale_Condition_Normal	8367.5845	0.1
Sale_Condition_Partial	12015.2260	0.1
Sale_Condition_other	7677.5821	0.1

---

From the regression output above, we can see lots of very small coefficients attached to the majority of our available predictors.

## Implementing the LASSO

The work required to construct a LASSO model is nearly identical. We simply use `mixture = 1` in the model specification to signal that we want to use the LASSO constraint instead of the ridge constraint.

```
lasso_reg_spec <- linear_reg(mixture = 1, penalty = 0.1) %>%
  set_engine("glmnet")

lasso_reg_wf <- workflow() %>%
  add_model(lasso_reg_spec) %>%
  add_recipe(reg_rec)

lasso_reg_cv <- lasso_reg_wf %>%
  fit_resamples(ames_folds)

lasso_reg_cv %>%
```

```

collect_metrics() %>%
kable() %>%
kable_styling(bootstrap_options = c("striped", "hover"))

```

.metric	.estimator	mean	n	std_err	.config
rmse	standard	3.245268e+04	5	3148.2028813	Preprocessor1_Model1
rsq	standard	8.411357e-01	5	0.0204318	Preprocessor1_Model1

```

lasso_reg_cv %>%
  collect_metrics(summarize = FALSE) %>%
  kable() %>%
  kable_styling(bootstrap_options = c("striped", "hover"))

```

id	.metric	.estimator	.estimate	.config
Fold1	rmse	standard	3.992358e+04	Preprocessor1_Model1
Fold1	rsq	standard	8.145726e-01	Preprocessor1_Model1
Fold2	rmse	standard	3.363114e+04	Preprocessor1_Model1
Fold2	rsq	standard	8.363147e-01	Preprocessor1_Model1
Fold3	rmse	standard	2.462751e+04	Preprocessor1_Model1
Fold3	rsq	standard	8.892578e-01	Preprocessor1_Model1
Fold4	rmse	standard	3.834326e+04	Preprocessor1_Model1
Fold4	rsq	standard	7.819507e-01	Preprocessor1_Model1
Fold5	rmse	standard	2.573790e+04	Preprocessor1_Model1
Fold5	rsq	standard	8.835826e-01	Preprocessor1_Model1

```

lasso_reg_fit <- lasso_reg_wf %>%
  fit(ames_train)

lasso_reg_fit %>%
  tidy() %>%
  filter(estimate != 0) %>%
  kable() %>%
  kable_styling(bootstrap_options = c("striped", "hover"))

```

term	estimate	penalty
Intercept	204174.844499	0.1

Lot_Frontage	302.118404	0.1
Lot_Area	2652.964700	0.1
Year_Built	5610.573998	0.1
Year_Remod_Add	4367.645860	0.1
Mas_Vnr_Area	4482.646323	0.1
BsmtFin_SF_1	-738.656408	0.1
BsmtFin_SF_2	736.862775	0.1
Bsmt_Unf_SF	-2866.828195	0.1
Total_Bsmt_SF	5568.968315	0.1
Second_Flr_SF	6684.251469	0.1
Low_Qual_Fin_SF	-1166.224044	0.1
Gr_Liv_Area	25928.430006	0.1
Bsmt_Full_Bath	2439.591819	0.1
Bsmt_Half_Bath	-621.816279	0.1
Full_Bath	3013.901185	0.1
Half_Bath	1058.224436	0.1
Bedroom_AbvGr	-1847.998427	0.1
Kitchen_AbvGr	-2158.528555	0.1
TotRms_AbvGrd	1215.142354	0.1
Fireplaces	6293.765983	0.1
Garage_Cars	7167.390067	0.1
Garage_Area	-18.026412	0.1
Wood_Deck_SF	1106.285872	0.1
Open_Porch_SF	-916.391841	0.1
Enclosed_Porch	1663.492292	0.1
Three_season_porch	456.706922	0.1
Screen_Porch	3677.763138	0.1
Pool_Area	-2354.835149	0.1
Misc_Val	-5627.001608	0.1
Mo_Sold	-572.996463	0.1
Year_Sold	-1369.021517	0.1
Longitude	1733.166334	0.1
Latitude	5384.510158	0.1
MS_SubClass_One_Story_1945_and_Older	-4308.968247	0.1
MS_SubClass_One_Story_1946_and_Newer_All_Styles	1678.308280	0.1
MS_SubClass_One_Story_PUD_1946_and_Newer	-1480.098139	0.1
MS_SubClass_Two_Story_1946_and_Newer	32.282767	0.1
MS_SubClass_other	6322.081252	0.1
MS_Zoning_Residential_Medium_Density	-6101.997383	0.1

MS_Zoning_other	-9260.875155	0.1
Street_other	-22970.598540	0.1
Alley_other	-7.656359	0.1
Lot_Shape_Slightly_Irregular	1771.961285	0.1
Lot_Shape_other	1838.909134	0.1
Land_Contour_other	-1349.950041	0.1
Utilities_other	-9062.911216	0.1
Lot_Config_CulDSac	10270.569626	0.1
Lot_Config_Inside	565.003677	0.1
Lot_Config_other	-6278.376017	0.1
Land_Slope_other	4625.142399	0.1
Neighborhood_Edwards	-10973.471669	0.1
Neighborhood_Gilbert	-19530.932784	0.1
Neighborhood_North_Ames	-8414.765814	0.1
Neighborhood_Northridge_Heights	14589.995488	0.1
Neighborhood_Old_Town	-11134.705591	0.1
Neighborhood_Somerset	8635.024580	0.1
Neighborhood_other	2810.367615	0.1
Condition_1_Norm	12745.603198	0.1
Condition_1_other	4603.785551	0.1
Condition_2_other	-1162.412692	0.1
Bldg_Type_TwnhsE	-18596.435680	0.1
Bldg_Type_other	-17463.487230	0.1
House_Style_One_Story	10066.653129	0.1
House_Style_Two_Story	-8523.299507	0.1
House_Style_other	-2571.006941	0.1
Overall_Qual_Average	-1436.679065	0.1
Overall_Qual_Below_Average	-2595.730854	0.1
Overall_Qual_Good	2280.491136	0.1
Overall_Qual_Very_Good	15680.512980	0.1
Overall_Qual_other	26471.108719	0.1
Overall_Cond_Average	-3237.595266	0.1
Overall_Cond_Good	4664.798807	0.1
Overall_Cond_other	-3679.502049	0.1
Roof_Style_Hip	3976.513691	0.1
Roof_Style_other	-5087.445097	0.1
Exterior_1st_MetalSd	3681.733444	0.1
Exterior_1st_Plywood	1478.941547	0.1
Exterior_1st_VinylSd	1114.089194	0.1
Exterior_1st_Wd.Sdng	-137.300127	0.1

Exterior_1st_other	7548.520118	0.1
Exterior_2nd_Plywood	-4165.802943	0.1
Exterior_2nd_VinylSd	702.891204	0.1
Exterior_2nd_Wd.Sdng	3609.641444	0.1
Exterior_2nd_other	-3116.316461	0.1
Mas_Vnr_Type_None	7630.970285	0.1
Mas_Vnr_Type_Stone	2241.854107	0.1
Mas_Vnr_Type_other	-6568.695833	0.1
Exter_Qual_Typical	-5426.837474	0.1
Exter_Qual_other	10846.988450	0.1
Exter_Cond_Typical	-1067.575915	0.1
Exter_Cond_other	-13298.737373	0.1
Foundation_PConc	4605.195066	0.1
Bsmt_Qual_Good	-23745.004600	0.1
Bsmt_Qual_Typical	-24768.334851	0.1
Bsmt_Qual_other	-25365.513028	0.1
Bsmt_Cond_other	-2662.979330	0.1
Bsmt_Exposure_Gd	12971.464880	0.1
Bsmt_Exposure_Mn	-7258.496092	0.1
Bsmt_Exposure_No	-9341.789381	0.1
Bsmt_Exposure_other	-9581.993387	0.1
BsmtFin_Type_1_BLQ	364.916614	0.1
BsmtFin_Type_1_GLQ	6708.401065	0.1
BsmtFin_Type_1_LwQ	-4053.082751	0.1
BsmtFin_Type_1_Unf	-124.232722	0.1
BsmtFin_Type_2_other	-1766.478865	0.1
Heating_other	-2679.276782	0.1
Heating_QC_Good	-2153.910215	0.1
Heating_QC_Typical	-5213.089203	0.1
Heating_QC_other	-10407.381099	0.1
Central_Air_Y	2548.570117	0.1
Electrical_SBrkr	624.529636	0.1
Electrical_other	851.058235	0.1
Kitchen_Qual_Good	-24550.199978	0.1
Kitchen_Qual_Typical	-27862.653516	0.1
Kitchen_Qual_other	-30664.715311	0.1
Functional_other	-16889.690718	0.1
Fireplace_Qu_No_Fireplace	4546.031514	0.1
Fireplace_Qu_Typical	-1105.453354	0.1
Fireplace_Qu_other	1172.042634	0.1



Garage_Type_BuiltIn	-5182.566720	0.1
Garage_Type_Detchd	-1238.501946	0.1
Garage_Type_other	-10324.464210	0.1
Garage_Finish_RFn	-2806.603971	0.1
Garage_Finish_Unf	-969.339686	0.1
Garage_Qual_Typical	-2427.037740	0.1
Garage_Qual_other	1761.572629	0.1
Garage_Cond_other	-9021.722065	0.1
Paved_Drive_Paved	3943.712969	0.1
Paved_Drive_other	6114.141994	0.1
Pool_QC_other	17880.143965	0.1
Fence_No_Fence	-2711.857292	0.1
Fence_other	-1542.616891	0.1
Misc_Feature_other	9460.842093	0.1
Sale_Type_WD	-124.694469	0.1
Sale_Type_other	-1217.105822	0.1
Sale_Condition_Normal	11499.733894	0.1
Sale_Condition_Partial	18556.312374	0.1
Sale_Condition_other	11574.716158	0.1

---

We can see in the LASSO model that many of the predictors were assigned coefficients of 0. The LASSO procedure identified that those predictors weren't worth "buying", so it left them out and only included the most useful predictors!

## Summary

In this notebook, we introduced two new regression models: *Ridge Regression* and the *LASSO*. These models are of the familiar form  $\mathbb{E}[y] = \beta_0 + \beta_1x_1 + \beta_2x_2 + \dots + \beta_kx_k$ , but are fit using a *constrained* optimization procedure rather than *ordinary least squares*. When we implement Ridge and the LASSO, we are reducing the likelihood that the resulting model is overfit. With these models, however, we must remember that all numerical predictors need to be *scaled* and no missing data can be present in the training data.