Applied Machine Learning - Feature Engineering and Preprocessing

Max Kuhn (RStudio)

Load Packages

Load Ames Housing Data

Preprocessing and Feature Engineering

This part mostly concerns what we can to our variables to make the models more effective.

This is mostly related to the predictors. Operations that we might use are:

- transformations of individual predictors or groups of variables
- alternate encodings of a variable
- elimination of predictors (unsupervised)

In statistics, this is generally called the data. As usual, the computer science side of modeling has a much flashier name: .

Reasons for Modifying the Data

- Some models (-NN, SVMs, PLS, neural networks) require that the predictor variables have the same units. and the predictors can be used for this purpose.
- Other models are very sensitive to correlations between the predictors and or can improve the model.
- As we'll see in an example, changing the scale of the predictors using a can lead to a big improvement.
- In other cases, the data can be in a way that maximizes its effect on the model. Representing the date as the day of the week can be very effective for modeling public transportation data.
- Many models cannot cope with missing data so strategies might be necessary.
- Development of new that represent something important to the outcome (e.g. compute distances to public transportation, university buildings, public schools, etc.)

Preprocessing Categorical Predictors

Dummy Variables

One common procedure for modeling is to create numeric representations of categorical data. This is usually done via : a set of binary 0/1 variables for different levels of an R factor.

For example, the Ames housing data contains a predictor called with levels: 'Gravel', 'No_Alley_Access', 'Paved'.

Most dummy variable procedures would make numeric variables from this predictor that are 1 when the observation has that level, and 0 otherwise.

Gravel	0	0
No_Alley_Access	1	0
Paved	0	1

Dummy Variables

If there are levels of the factor, only -1 dummy variables are created since the last can be inferred from the others. There are different contrast schemes for creating the new variables.

For ordered factors, contrasts are used. See this blog post for more details.

How do you create them in R?

The formula method does this for you[1]. Otherwise, the traditional method is to use to create a matrix. However, there are some caveats to this that can make things difficult.

We'll show another method for making them shortly.

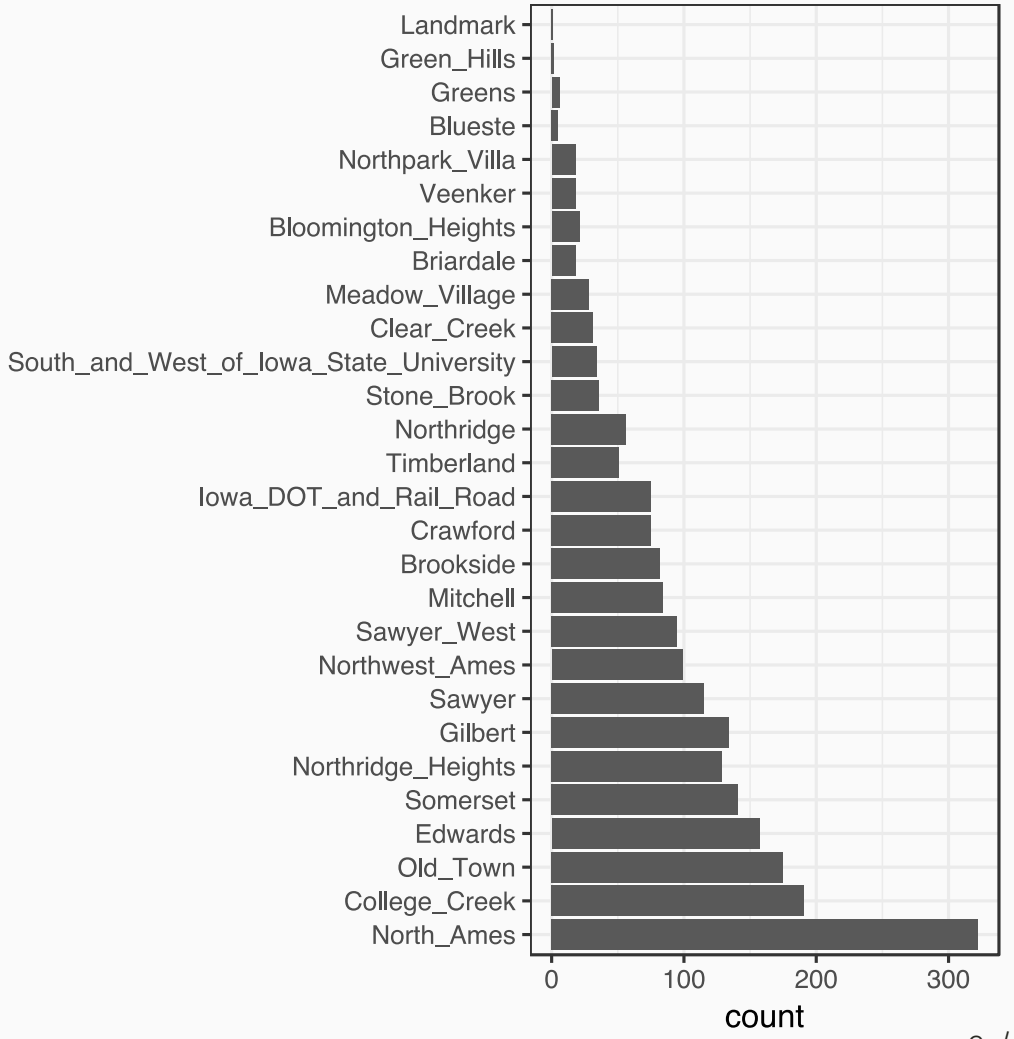
[1] at least. Tree- and rule-based model functions do not. Examples are , , , , , and others.

Infrequent Levels in Categorical Factors

One issue is: what happens when there are very few values of a level?

Consider the Ames training set and the variable.

If these data are resampled, what would happen to Landmark and similar locations when dummy variables are created?



Infrequent Levels in Categorical Factors

A predictor that has only a single value (zero) would be the result.

Many models (e.g. linear/logistic regression, etc.) would find this numerically problematic and issue a warning and values for that coefficient. Trees and similar models would not notice.

There are two main approaches to dealing with this:

- Run a filter on the training set predictors prior to running the model and remove the zero-variance predictors.
- Recode the factor so that infrequently occurring predictors (and possibly new values) are pooled into an "other" category.

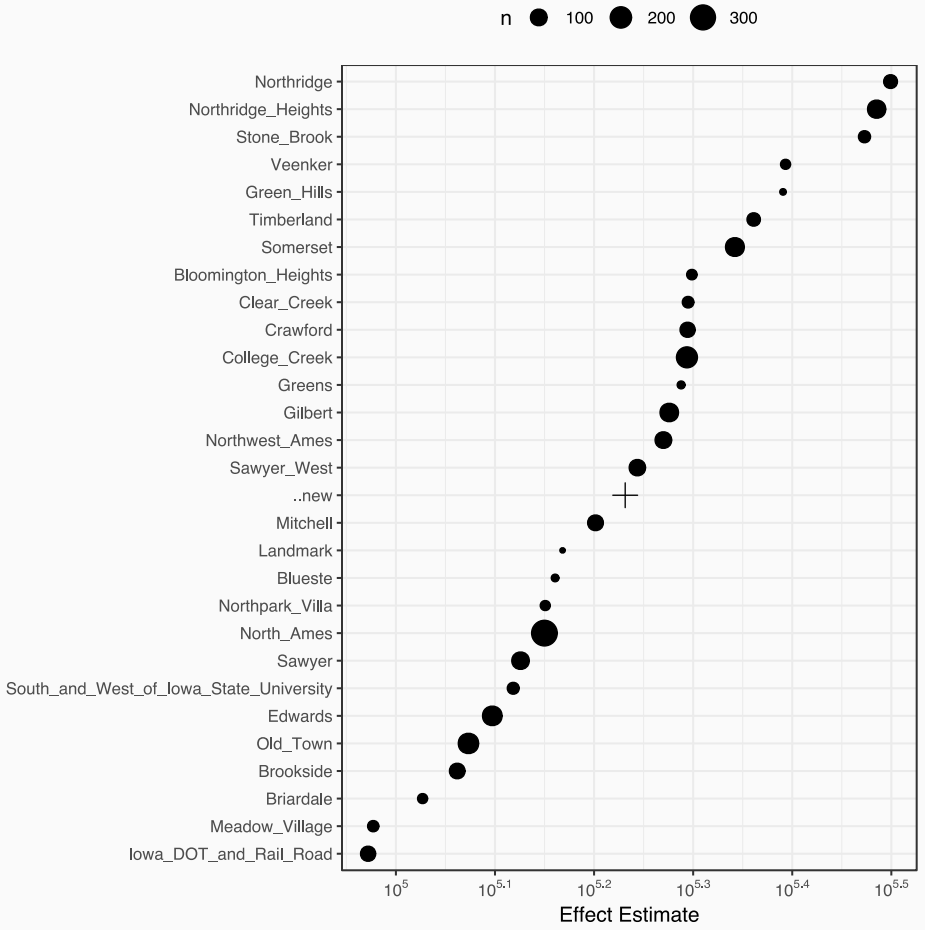
However, and the formula method are incapable of doing either of these.

Other Approaches

A few other approaches that might help here:

- or of categorical predictors estimate the mean effect of the outcome for every factor level in the predictor. These estimates are used in place of the factor levels. Shrinkage/regularization can improve these approaches.
- use a neural network to create features that capture the relationships between the categories and the outcome.

An add-on to called can be used for these encodings.



Recipes

Recipes are an alternative method for creating the data frame of predictors for a model.

They allow for a sequence of that define how data should be handled.

Recall the previous part where we used the formula ? These steps are:

- Assign to be the outcome
- Assign and as predictors
- Log transform the outcome

To start using a recipe, these steps can be done using

```
```

This creates the recipe for data processing (but does not execute it yet)

Recipes and Categorical Predictors

To deal with the dummy variable issue, we can expand the recipe with more steps:

Note that we can use standard selectors as well as some new ones based on the data type ( ) or by their role in the analysis ( ).

Using Recipes

Preparing the Recipe

Now that we have a preprocessing , let's run it on the training set to the recipe:

```

```

```

```

Here, the "training" is to determine which levels to lump together and to enumerate the factor levels of the variable.

An unused option, , that keeps the processed version of the training set around so we don't have to recompute it.

Preparing the Recipe

Getting the Values

Once the recipe is prepared, it can be applied to any data set using :

If , the training set does not need to be "rebaked". The function can return the processed version of the training data.

Selectors can be used with to only extract relevant columns and the default is .

How Data Are Used

Note that we have:

In the first case, is used by the function only to determine the column names and types (e.g. factor, numeric, etc). A small subset can be passed via or .

For , the argument should have all of the data used to estimate parameters and other quantities.

For , is the data that the pre-processing should be applied to.

Hands-On: Zero-Variance Filter

Instead of using , take 10 minutes and research how to eliminate any zero-variance predictors using the reference site.

Re-run the recipe with this step.

What were the results?

Do you prefer either of these approaches to the other?

Principal Component Analysis

A Bivariate Example

The plot on the right shows two predictors from a real set where the objective is to predict the two classes.

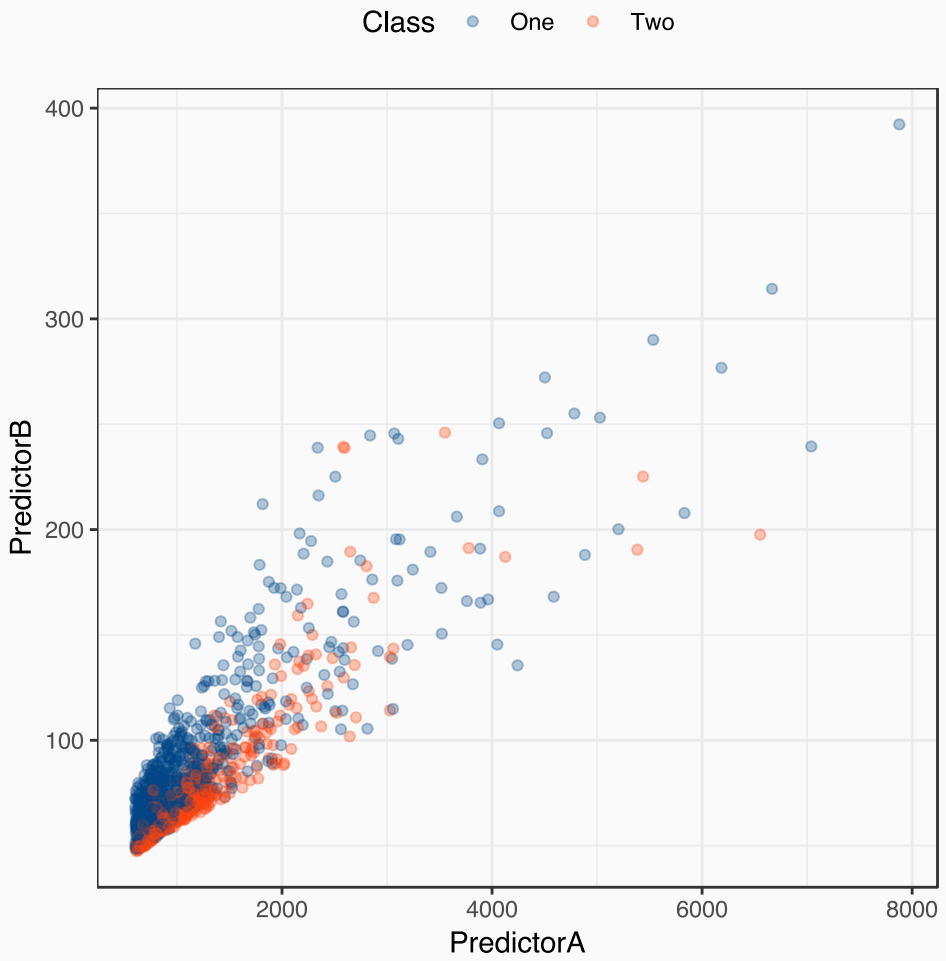
The predictors are strongly correlated and each has a right-skewed distribution.

There appears to be some class separation but only in the bivariate plot; the individual predictors show poor discrimination of the classes.

Some models might be sensitive to highly correlated and/or skewed predictors.

Is there something that we can do to make the predictors ?

?



A Bivariate Example

We might start by estimating transformations of the predictors to resolve the skewness.

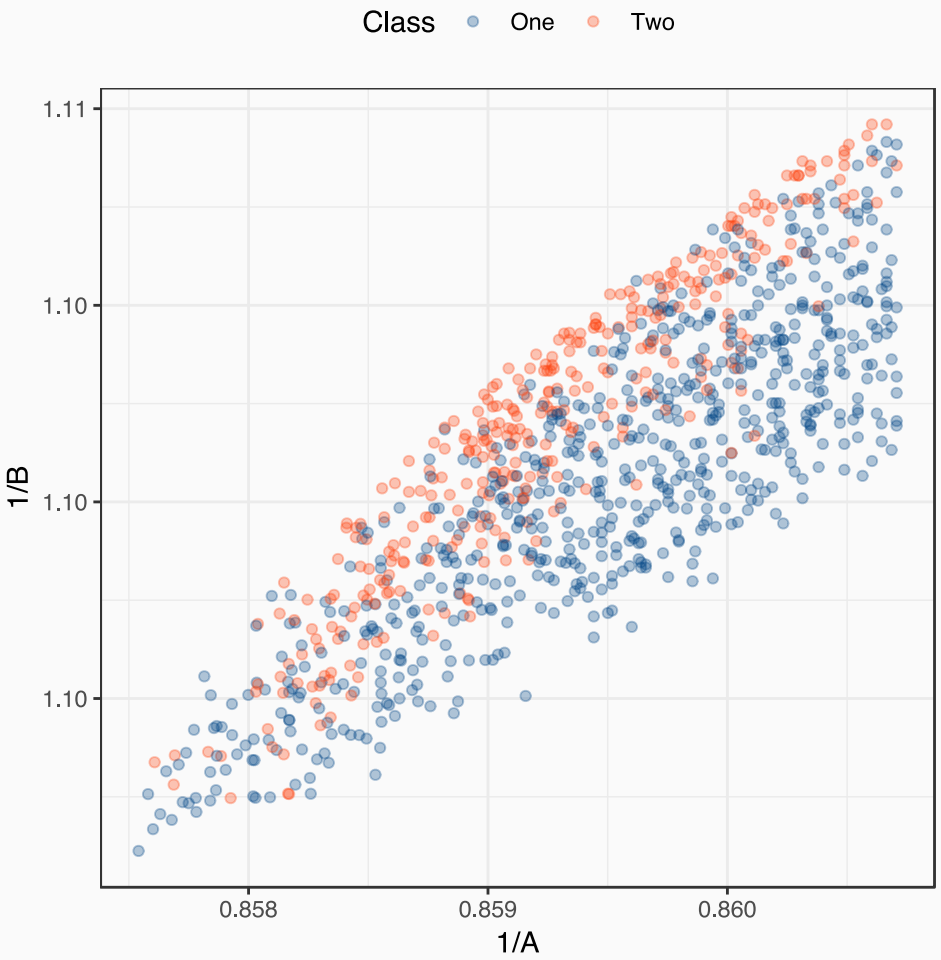
The Box-Cox transformation is a family of transformations originally designed for the outcomes of models. We can use it here for the predictors.

It uses the data to estimate a wide variety of transformations including the inverse, log, sqrt, and polynomial functions.

Using each factor in isolation, both predictors were determined to need inverse transformations (approximately).

The figure on the right shows the data after these transformations have been applied.

A logistic regression model shows a substantial improvement in classifying using the altered data.



More Recipe Steps

The package has a rich set of steps that can be used including transformations, filters, variable creation and removal, dimension reduction procedures, imputation, and others.

For example, in the previous bivariate data problem, the Box-Cox transformation was conducted using:

Correlated Predictors

In the Ames data, there are potential clusters of :

- proxies for size: , , , , etc.
- quality fields: , , , , etc.

It would be nice if we could combine/amalgamate the variables in these clusters into a single variable that represents them.

Another way of putting this is that we would like to create artificial features of the data that account for a certain amount of variation in the data.

There are a few different methods that can accomplish this; we will focus on principal component analysis (PCA) as a solution. Another, regularization, will be discussed later.

PCA Signal Extraction

Principal component analysis (PCA) is a multivariate statistical technique that can be used to create artificial new variables from an existing set.

The new variables are created to account for the most variation in the data. In this case, "variation" means correlation.

Conceptually, PCA determines which variables account for the most correlation in the data and creates a new variable that is a linear combination of all the predictors.

- This is called the (aka).
- This linear combination emphasizes the variables that are the most correlated.

The information that PC #1 represents is then .

The second PCA component is the linear combination that accounts for the most left-over correlation in the data (and so on).

PCA Signal Extraction

To recap:

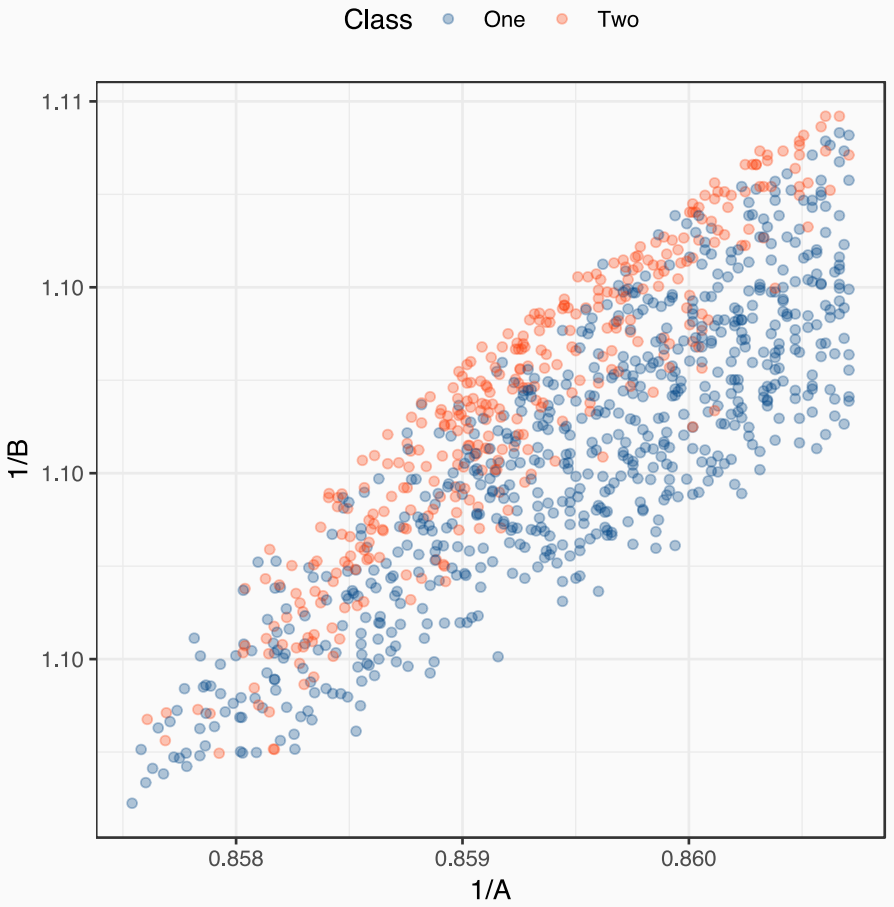
- The components account for as much as the variation in the original data as possible.
- Each component is uncorrelated with the others.
- The new variables are of all of the input variables and are effectively unitless (It is generally a good idea to center and scale your predictors because of this).

For our purposes, we would use PCA on the to:

- Reduce the number of variables exposed to the model (but this is not feature selection).
- Combat excessive correlations between the predictors (aka multicollinearity).

In this way, the procedure is often called but this is poorly named since there is no guarantee that the new variables will have an association with the outcome.

Back to the Bivariate Example - Transformed Data



Back to the Bivariate Example - Recipes

We can build on our transformed data recipe and add normalization:

Back to the Bivariate Example

Recall that even after the Box-Cox transformation was applied to our previous example, there was still a high degree of correlation between the predictors.

After the transformation, the predictors were centered and scaled, then PCA was conducted. The plot on the right shows the results.

Since these two predictors are highly correlated, the first component captures 91.7% of the variation in the original data. However...

...recall that PCA does not guarantee that the components are associated with the outcome. In this example, the component has the association with the outcome.

Resampling and Preprocessing

It is important to realize that almost all preprocessing steps that involve estimation should be bundled inside of the resampling process so that the performance estimates are not biased.

- : preprocess the data, resample the model
- : resample the preprocessing and modeling

Also:

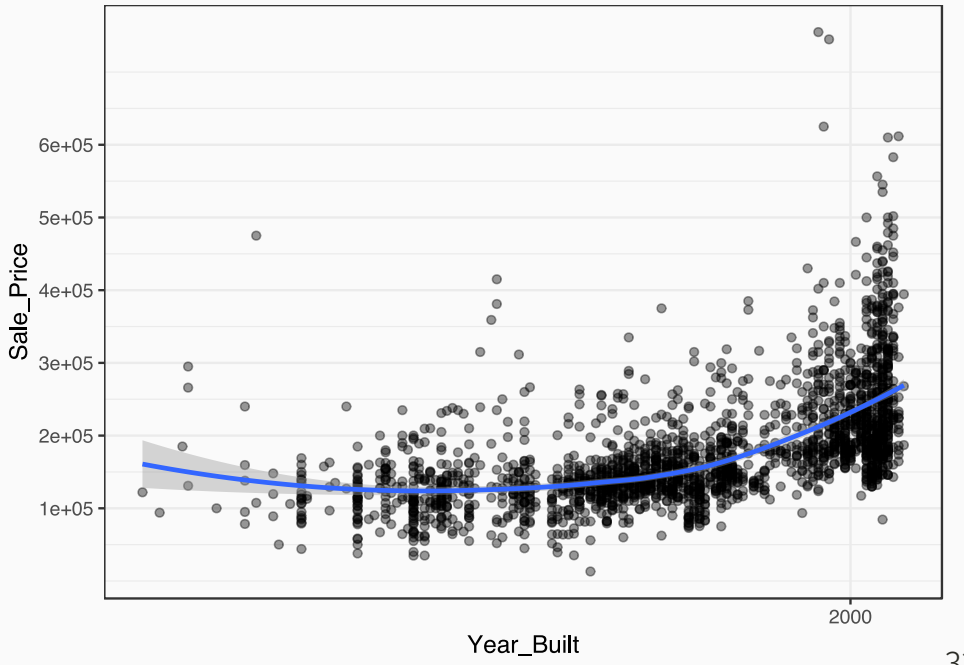
- Avoid by having all operations in the modeling process occur only on the training set.
- Do not reestimate anything on the test set. For example, to center new data, the training set mean is used.

Interaction Effects

Interactions

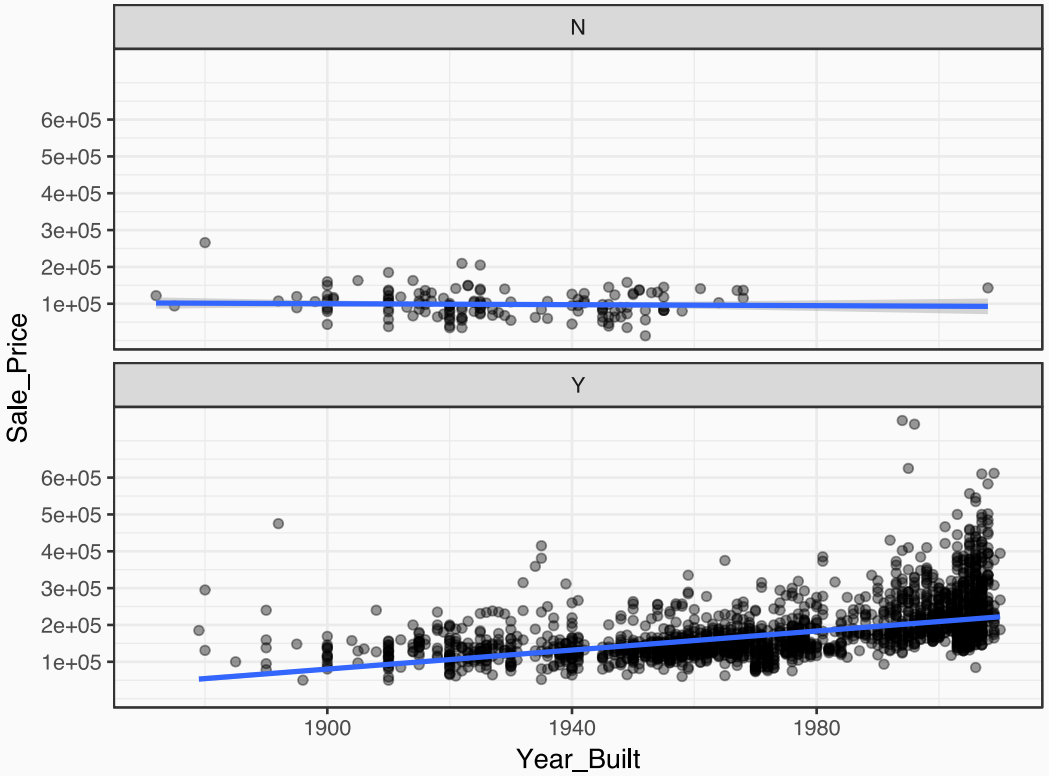
An between two predictors indicates that the relationship between the predictors and the outcome cannot be describe using only one of the variables.

For example, let's look at the relationship between the price of a house and the year in which it was built. The relationship appears to be slightly nonlinear, possibly quadratic:



Interactions

However... what if we separate this trend based on whether the property has air conditioning (93.4% of the training set) or not (6.6%):



Interactions

It appears as though the relationship between the year built and the sale price is somewhat for the two groups.

- When there is no AC, the trend is perhaps flat or slightly decreasing.
- With AC, there is a linear increasing trend or is perhaps slightly quadratic with some outliers at the low end.

Interactions in Recipes

We first create the dummy variables for the qualitative predictor ( ) then use a formula to create the interaction using the operator in an additional step:

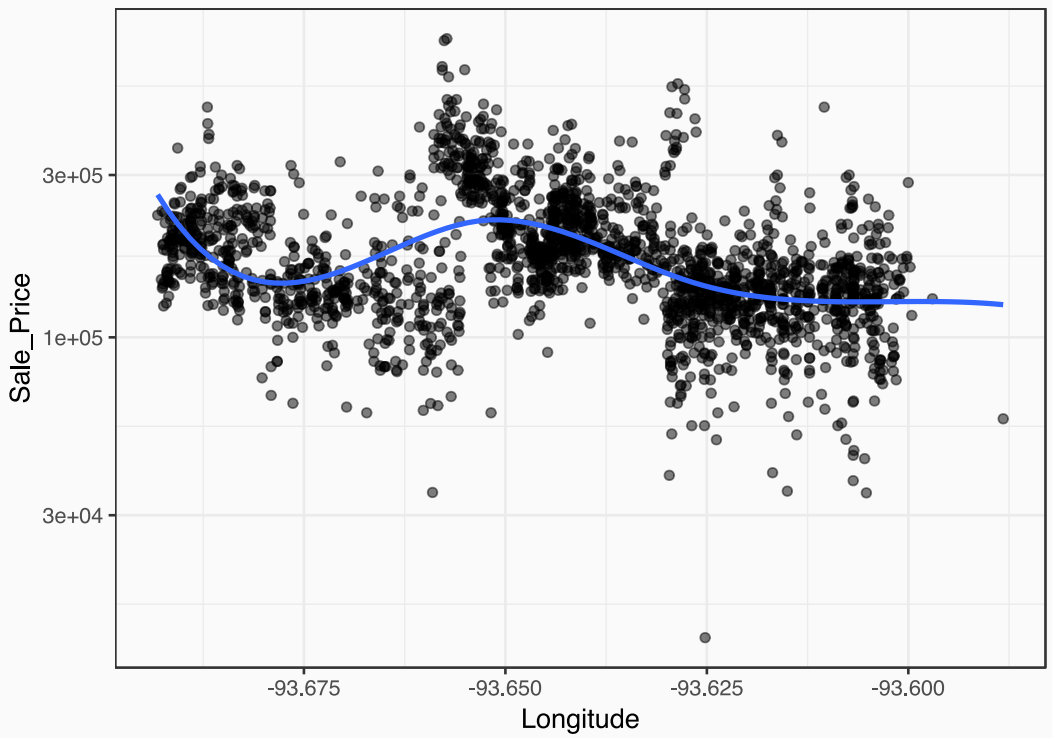
Adding Recipes to our Workflows

Linear Models Again

Let's add a few extra predictors and some preprocessing.

- Two numeric predictors are very skewed and could use a transformation (and).
- We'll add neighborhood in as well and a few other house features.
- The -NN model suggests that the coordinates can be helpful but probably require a nonlinear representation. We can add these using with 5 degrees of freedom. To evaluate this, we will create two versions of the recipe to evaluate this hypothesis.

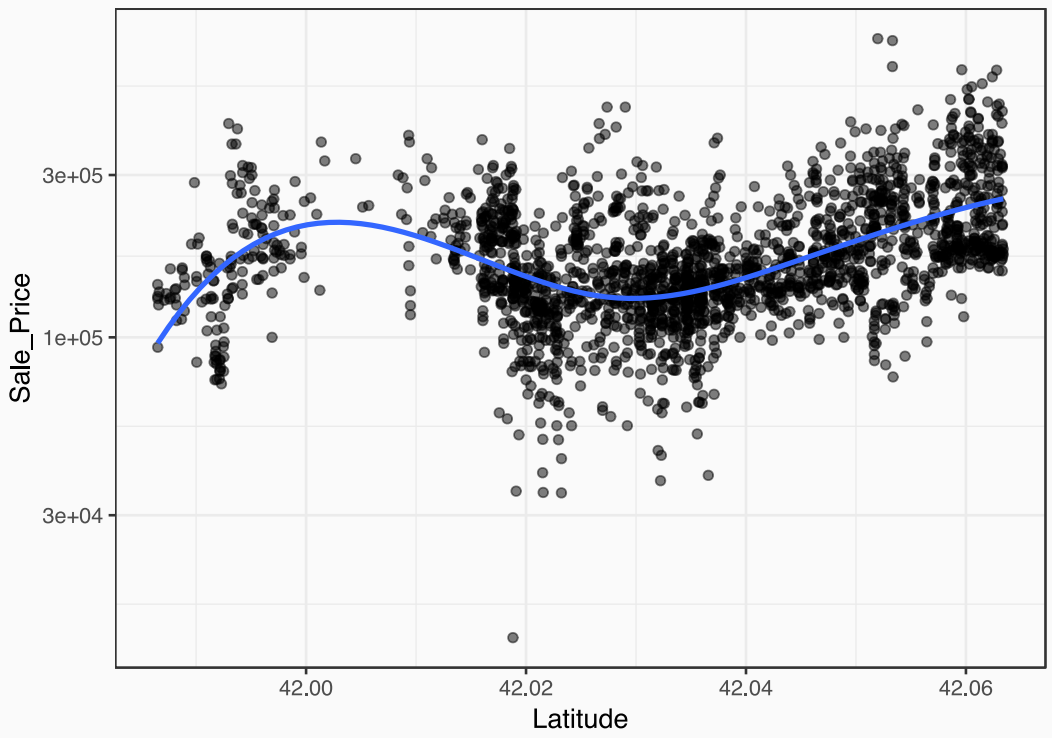
Longitude



Splines add nonlinear versions of the predictor to a linear model to create smooth and flexible relationships between the predictor and outcome.

This "basis expansion" technique will be seen again in the regression section of the workshop.

Latitude



Preparing the Recipes

Our first step is the run on the recipe but using each of the analysis sets.

has a function that is a wrapper around that can be used to map over the split objects, prepping on the analysis set of each one:

Fitting the Models

We can use code that is very similar to the previous section.

This code will use the recipe object to get the data. Since each analysis set is used to train the recipe, our previous use of means that the processed version of the data is within the recipe. This can be returned via the function.

Predicting the Assessment Set

This is a little more complex. We need three elements contained in our tibble:

- the split object (to get the assessment data)
- the recipe object (to process the data)
- the linear model (for predictions)

The function is not too bad:

Predicting the Assessment Set

Since we have three inputs, we will use to walk along all three columns in the tibble.

We do get some warnings that the assessment data are outside the range of the analysis set values:

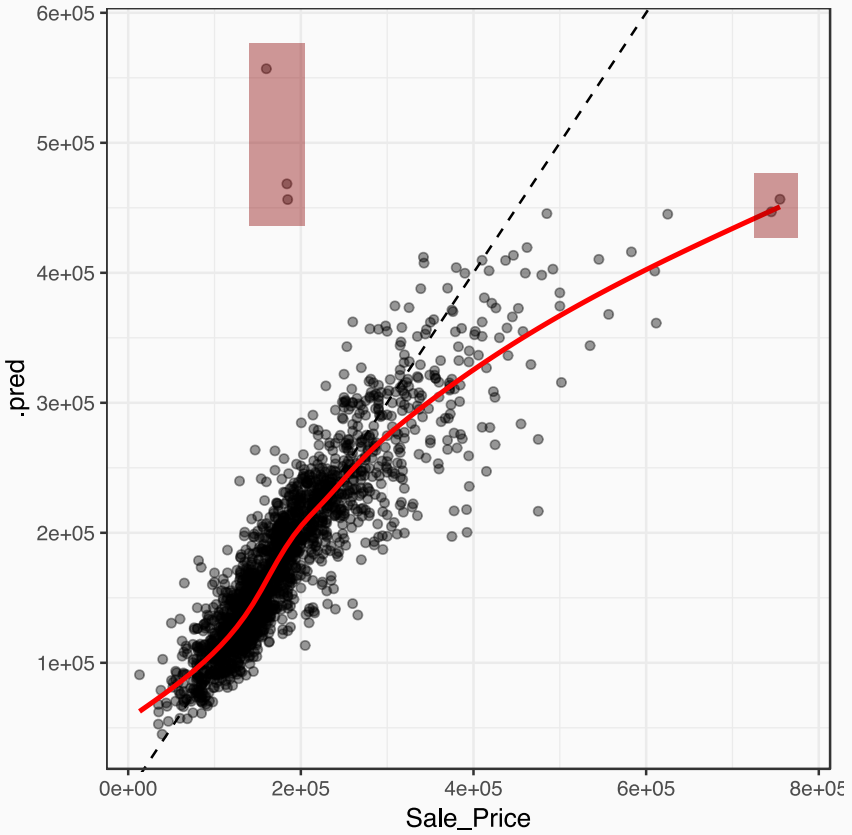
Predicting the Assessment Set

will compute a small set of summary statistics for the model based on the type of outcome (e.g. regression, classification, etc).

These results are better than our -NN model but "

".

Graphical Checks for Fit



- The code used for highlighting the points is slightly more complex

Graphical Checks for Fit

The current model is:

- Drastically -predicting the price of three houses.
- Significantly -predicting the price of a number of expensive houses.

We would we do next?

1. Try to understand the big residuals. Are these aberrant houses or does this have something to do with our model? Maybe those extrapolation warnings?
2. Find more predictors that differentiate the more expensive houses with the others.
3. Try a different model.

(I wrote this slide long before the next one)

About Those Five Houses

From Dmytro Perepolkin via :

> I did a little bit of research on Kaggle regarding those five houses (three "partial sale" houses in Edwards and two upscale in Northridge):
>
> >

Sometimes it really pays off to be a forensic statistician.