Case Study Business Report

Tyler White, Ford Higgins, Ryan Campa, & Yiqiang Zhao

10/5/2017

Explanatory

1. Feature Selection and Imputation

The data provided on the Ames, Iowa housing market is not very clean to begin with, and includes several variables that can be cut out of the dataset immediately for intuitive reasons.

The `Id` variable represents the row number within the data set, and has no relation to the actual data on housing. We threw it out because it can neither cause or predict future points, nor explain existing data.

We removed variables whose entries were nearly all the same value, including `MiscFeature` (96.3% of observations), `PoolQC` (99.5%), and `Utilities` (99.9%).

Following this intuitive cleaning, we removed `MSZoning`, `Exterior2nd`, `GarageYrBlt`, and `SaleType` due to their high collinearity, which was found using VIF and trial and error. These removals improved the overall VIF scores to the point we can continue without worrying about collinearity popping up while modeling.

In this data set the NA values are almost always meaningful. An example is that if a house does not have a basement, it is assigned a `BsmtQual` of `NA`. We simply add a new factor for `None` when dealing with categorical variables so the regression models do not ignore these data points. We replace the `NA`s present in continuous columns of data with sensible values, such as setting `GarageArea` to 0 for houses without a garage. For continuous data there is a sensible value to put replace `NA`s with. An example is GarageArea is 0 if you do not have a garage.

The only continuous data we use imputation on is `LotFrontage`, where we created a linear model between `LotFrontage` and `LotArea` since there are no `NA`s in the `LotArea` column. We then predicted values for `LotFrontage` when it was `NA` using their corresponding `LotArea` and the linear model. The imputation improved the R^2 of the model without adding much collinearity.

We also noted that many variables were rankings, which are loaded as categories by default. We change these to be on a scale of 1 to 5 scale instead of Poor, Fair, Typical, Good, or Excellent. This switch improves the model's interpretability.

2. K-Means X-Outlier detection

We discovered k-means can tell us when a point will be poorly predicted based on the X values for this new data point. We found that k-means groups ~98% of the data into a single cluster while the rest of the data is divided into two clusters, which is shown in the table below.

Cluster ID	Number of points
1	716
2	10
3	4

These k-means outlier groups are very far from the main cluster, as determined by the L_2 metric and shown in the following table.

Cluster ID	Min Range	Max Range
1	469	16,600
2	18,100	61,200
3	105,500	205,600

As you can see, the points in clusters 2 and 3 are very far from the main group's centroid. Thus, we can conclude that clusters 2 and 3 are outliers in X. We can accurately predict when our model can actually predict new points it is given by determining their cluster. Typically ~1.7% of all new data points end up being put into one of the two outlier clusters.

New points detected as outliers have a typical MSPE of $2 * 10^{10}$ as compared to a MSPE of $4 * 10^8$ for the main cluster, showing that the points k-means identifies as outliers are usually predicted poorly.

Due to the scarcity of these points, we are unable to use regression techniques to predict the sale price of new points belonging to the outlier clusters. Instead, we use the mean sale price of the existing outliers for prediction on new points, and this also allows us to report when a new point is an outlier. This knowledge about a point could be useful, since it could tell us when an expert's opinion is necessary. We can also infer that if a point is not an outlier, then our predictions will be accurate.

3. Model

Our explanatory model follows the methodology below during training:

We perform variable selection first, as detailed above, before using the k-means outlier detection method.

We apply the Box-Cox transformation and determined $\lambda = \frac{1}{4}$ is a good choice for the data. This λ is a somewhat interpretable value that falls within the confidence interval produced by `Box-Cox`.

After the transformation, Lasso regression is used for another round of variable selection. We only keep βs that Lasso does not shrink to 0. We run an OLS regression with the reduced number of variables, and then use the fitted model as the input for DFFITS to find any influential points.

This is our final explanatory model for the points falling in the largest cluster of points found by k-means. However, we model the outlier points using their mean in the training data. We use this simple model due to how few points fall into those groups.

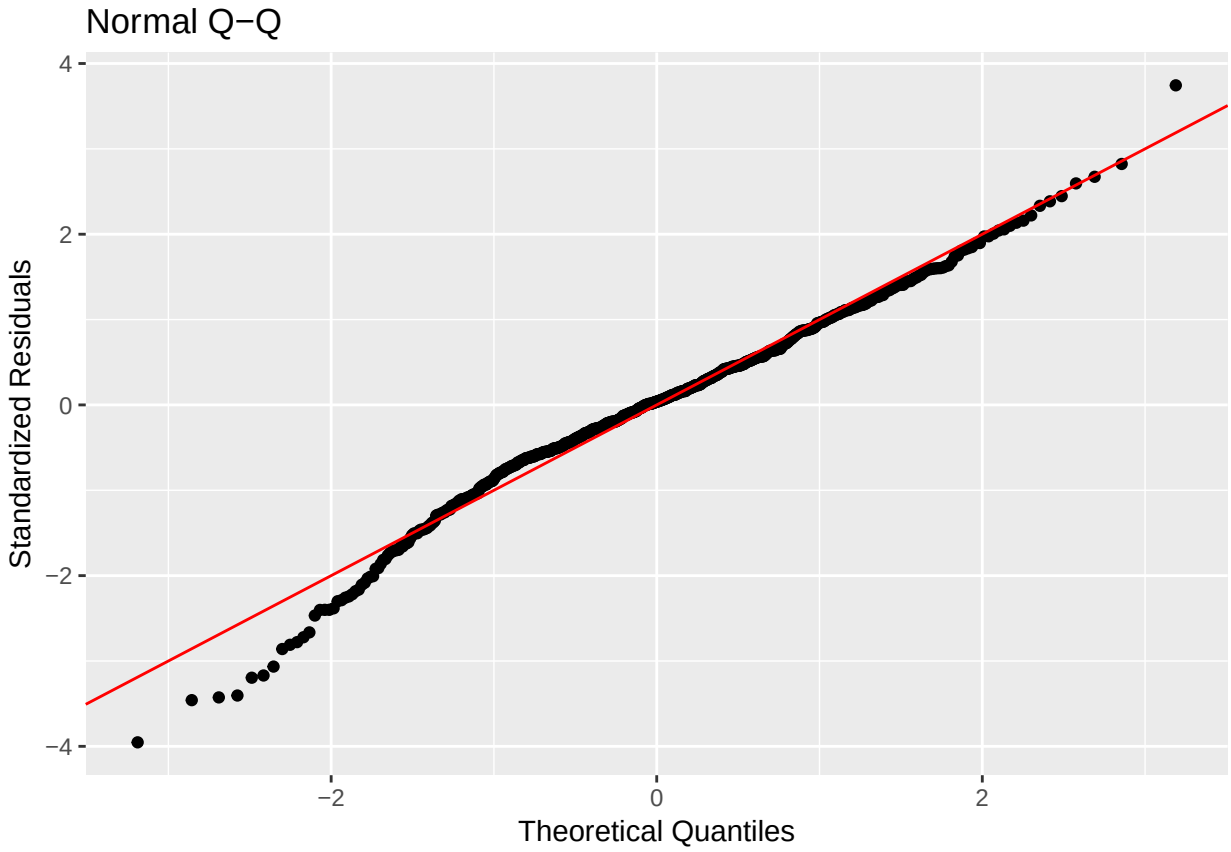
Predicting New Points:

Before predicting new points, we first determine which their correct k-means training cluster. If they are closest to the main group, we use our OLS regression model to predict the new house's sale price. If the point is in one of the outlier groups, we predict using the mean of that cluster. We report the MSPE of the main group points and the outliers separately, as well as the rate of outliers. We return the combined MSPE, too.

4. Checking Assumptions

We use the Breusch–Pagan test to check for homoscedasticity and find a p-value of 0.5, indicating a 50% chance we would see data this extreme if homoscedasticity is present. From this result we conclude there is not evidence of heteroscedasticity.

We look at the qqplot of the standardized residuals to check if normality holds. We have plotted this below.



Note that the lower tail falls off from the line, so there may be concerns about the normality of our data.

The MSPE for our model is $4.4 * 10^8$ for the non-outlier points. The MSPE with the outliers is $7.8 * 10^8$. So are model is able to predict much better for the non-outlier group. We also found our model had an $R^2 = .95$, which indicates the model does a good job capturing the variability.

5. Results

We found that many common beliefs about important features for a house's sale price were upheld. Our model gives a prediction of $z = 4 * (price^{\frac{1}{4}} - 1)$, where z is the output of the Box-Cox transformation. It indicates a direct relationship between z and *price*. One of the largest factors for a house's price is its neighborhood, such as the Stone Bridge neighborhood, which has the highest prices. If a house is in Stone Bridge, then z increases by 3.23 with a p-value of $2.61 * 10^{-8}$ Which neighborhood the house is in is one of the biggest factor. Houses with brick faces and locations nears parks also cause the price to increase significantly.

Morty's house is placed in the main k-means cluster, so we can predict his house price well. The model predicts Morty could sell his house for \$163,600 with a 95% confidence interval between \$152,300 and \$175700 dollars.

Most of the important variables are largely based around the house's location and the type of sale. However, if he can improve the variable `OverallQual` and `OverallCond` from a 5 to a 7, as well as improving the `KitchenQual` from a `"TA"` to a `"GD"` his house should sell for much more.

If he implements these changes, we predict his house price will rise to \$196,600 with a 95% confidence interval between \$183,100 an \$210,700.

6. How to run the model

Run the script file attached and change the path `~/<FilePath>/<FileName>` to the correct one, on the line:

```
data <- read_csv("~/<FilePath>/<FileName>")
```

To view the statistically significant betas for z run the following code

```
View(b)
```

To test new data change file path `~/<FilePath>/<FileName>` to the correct path in:

```
new.data <- read_csv("~/<FilePath>/<FileName>")
```

The from predictNew is a list with 3 items. `prediction$predicted.y` is the values for all data points. `is.outlier` is a vector that tells if each point is a k-means outlier. `outlier.rate` is the rate that outliers were detected.

If you wish to test MSPE of just the non-outlier points run, where you replace `your.y.test` with your own test y.

```
mean((your.y.test[!prediction$is.outlier] - prediction$predicted.y[!prediction$is.outlier])^2)
```

For all the points combined run the following command:

```
mean((your.y.test - prediction$predicted.y)^2)
```

Prediction

1. Data Cleaning and Selection

After loading the dataset, we examined the data for incompleteness and redundancy. Similar to the methods described in the Explanatory section, we removed the following variables with a high percentage of a single value: `Alley`, `PoolQC`, `Fence`, `MiscFeature`, `ID`, `Utilities`, `GarageYrBlt`.

The following variables were converted to factors: `MSSubClass`, `MSZoning`.

2. Imputing

Several rows and columns contained a large number of `NA` values. After consulting the data description documentation, it appeared that several categorical variables used `NA` to denote 'missing' or 'None'. Since this was an issue with semantics, the following variables with `NA` were converted to 'None':

Variables Where NA set to 'None'		
GarageType	GarageQual	BsmtCond
GarageFinish	GarageCond	BsmtExposure
GarageCars	FireplaceQu	BsmtFinType1
GarageArea	BsmtQual	BsmtFinType2
MasVnrType		

There were instances where `NA` appeared in a column of a discrete variable. Upon further inspection, we decided to substitute 0 in place of `NA` because it was the most 'truthful' representation of the physical state. For instance, when a house did not include a garage, its value for `GarageArea` was `NA`. In this case, we replace `NA` with 0. The following variables with `NA` were replace with 0: `GarageArea`, `LotFrontage`, `MasVnrArea`.

After data cleaning and imputing, one row contained a single `NA` under the `Electrical` variable. We removed this single data point for convenience.

3. Feature Engineering

From personal experience, the neighborhood is often a good indicator of the general price range of a house. The neighborhood is often associated with a school district, public services, recreational activities, etc. Here in San Francisco, the cost of a one-bedroom condo in the Russian Hill neighborhood is much different than that of a one-bedroom condo in Dogpatch. To capture these sort of interactions, we permuted several pairs of categorical variables to create new features. In the case of `Neighborhood` and `BldgType`, a new `NeighborhoodBldgType` feature was created if more than 12 data points matched the new pair. For example, the `Edwards1Fam` feature was created because there were more than 12 houses which were located in the Edwards neighborhood and were single-family dwellings. The number 12 was chosen as the threshold because, after examining several pairs, it appeared to be a natural divide; values tended to be either much higher or lower than 12. The choice of pairs was influenced by results of the explanatory section as well as anecdotal evidence.

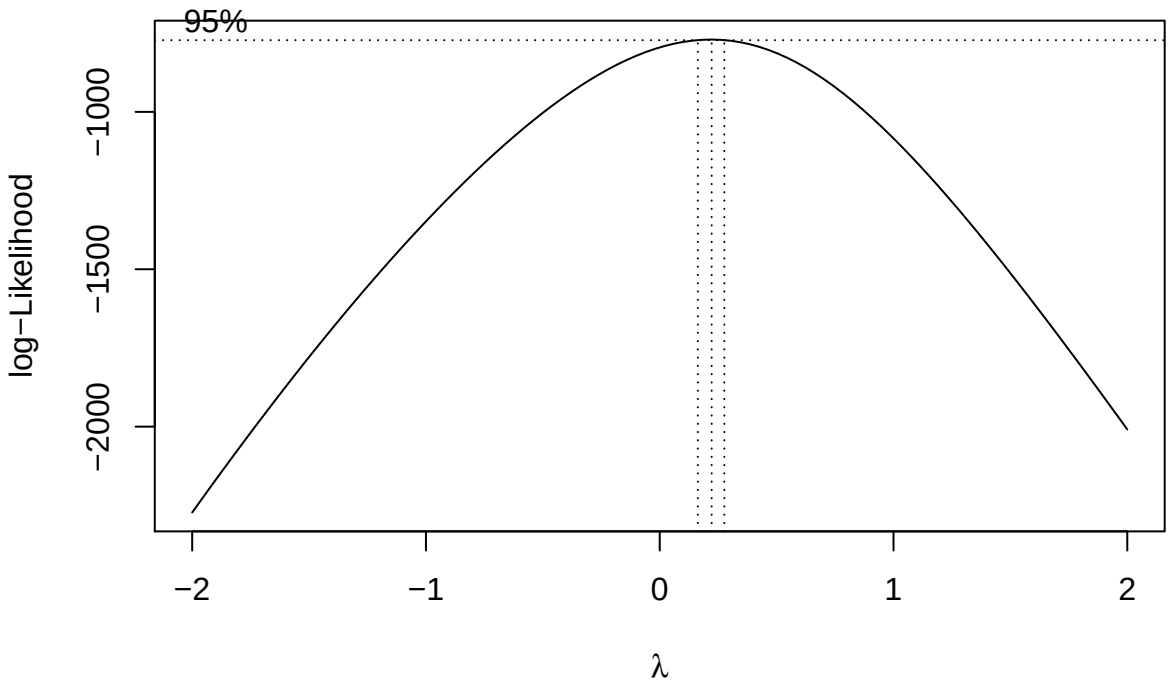
Table 4: New Features Pairs

Feature1	Feature2
Neighborhood	LotConfig
Neighborhood	BldgType
Neighborhood	HouseStyle
Neighborhood	Condition1
Neighborhood	SaleCondition
HouseStyle	BldgType
HouseStyle	SaleCondition
SaleType	SaleCondition
Exterior1st	Exterior2nd

4. Model

The Predictive Model does the following to train the model:
1) Use k-means method to determine the X outlier groups
2) Use Box-Cox transformation with $\lambda = \frac{2}{9}$
3) Use LASSO, Ridge, and Elastic-Net with alphas of 0.25, 0.5, 0.75

```
# Box-Cox Y transformation
powerLaw <- function(y, lambda) (y^lambda - 1)/lambda
# Box-Cox Y retransformation
inverseLaw <- function(z, lambda) (lambda * z + 1)^(1/lambda)
# Show the plot of the estimate value of lambda
MASS::boxcox(train.y ~ ., data = as.data.frame(train.x))
```



5. Discussion and Results

The k-means method of removing outliers allowed the model to do extremely well reducing MSPE for the clustered data points, at the cost of predicting outliers very poorly. We effectively created two models: one for the data points in the main cluster, and a second for the outliers. Because the outliers were few in number and dissimilar, we chose to estimate their Sale Price by using the mean Sale Price of the data set.

The results are tabulated below. "Good MSPE" represents the MSPE of our model when applied to the clustered data points. "Bad MSPE" represents the MSPE of our secondary model (the mean of the data set) when applied to the outliers. The Combined MSPE was calculated by combining the Good MSPE and the Bad MSPE. LASSO and Elastic Net had very similar Combined MSPEs and outperformed Ridge. The poor performance of Ridge may be due the sheer number of variables created by feature engineering. Elastic Net with alpha set at 0.5 performed the best by a small margin.

If more time was permitted, we would have explored more ideas for feature engineering in an effort to reduce the number of variables in the model. Visiting sites like Zillow or surveying home listings in the newspaper could provide insight into the most important features of a home.

	RIDGE	LASSO	Elastic.Net.0.25	Elastic.Net.0.50	Elastic.Net.0.75
good mspe	504753103	436188071	434030601	428762552	434391777
bad mspe	8869747243	8869747243	8869747243	8869747243	8869747243
combined mspe	818889719	751710380	749656861	744628676	749978777

6. Run the model

Make sure you put `Prediction Model.R` under the same directory as this file. The output will be good mspe, bad mspe, and combined mspe.

```
source("CS_Prediction_Model.R")

# the training date set
testfunc("<FilePath>/<Training Data FileName>", "<FilePath>/<Testing Data FileName>")
```