



## Project 1: Kaggle Challenge

Chrisitan Vega, Sammana Kabir, Nicholas Zermeno

California State University, Northridge

1839	<b>nickzermeno</b>		0.14377	10	3h
					
<p><b>Your Best Entry!</b> Your most recent submission scored 0.14377, which is an improvement of your previous score of 0.14870. Great job!</p>					
					<a href="#">Tweet this</a>

**Problem at Hand**

Given 79 explanatory variables with 1,460 entries of data, we are tasked with predicting the sale price of a residential home in Ames, Iowa on 1,458 different observations. We must use all that we have learned throughout the semester to put together a model that can most accurately predict what these prices will be given a testing set. The biggest problem is: this data isn't complete, and it sure is messy.

**Methods Used for Analysis**

The methodology for this project went something like this: Fix up the data -> Check for multicollinearity -> Address discrepancies in levels between datasets -> Build models -> Simplify models -> Run our predictions.

**Fixing the Data**

To begin to fix up the data, we had to first check to see what was missing in it. This was done using the function below. This allowed us the percent of data missing in each column.

```
na_count_train_test <- sapply(mice_train_full, function(y)
  sum(length(which(is.na(y)))))
na_count_train_test <- data.frame(na_count_train_test)
```

Now, some of these columns are categorical, so we went and changed the NA's that belonged to these columns to 0's so we can use the mice package imputation for the columns that were continuous. This ended up imputing the missing values for LotFrontage, MasVnrArea, and GarageYrBlt. Two categories stick out that don't use NA as a criteria, MasVnrType and Electrical. Both of these only have a few missing values so we use mode imputation in order to

complete them. A problem of mode imputation could be possible overfitting, so we had to remain cautious about this.

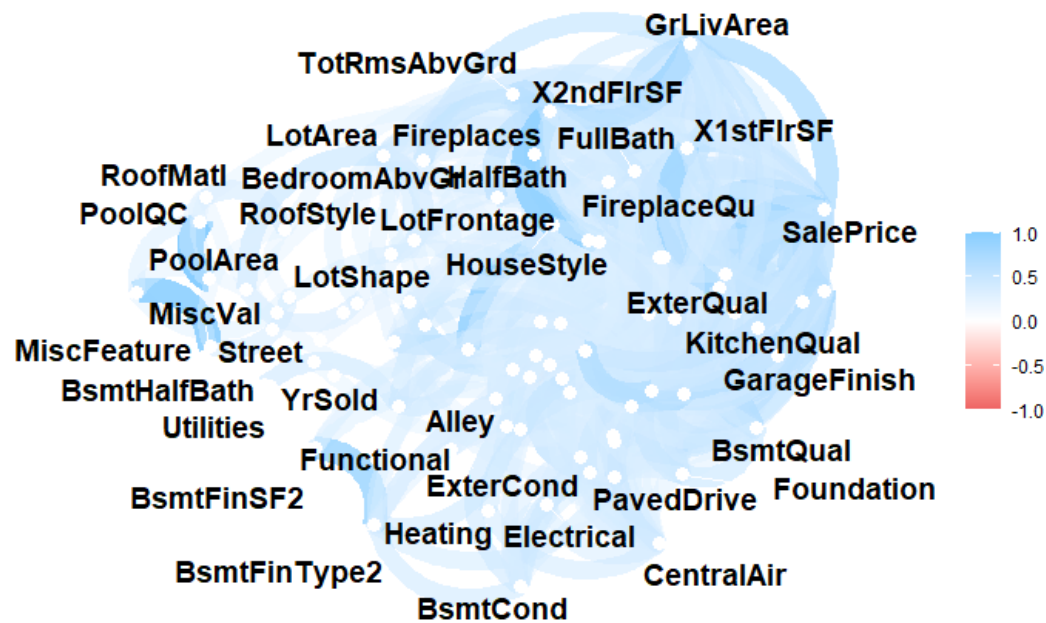
```
mice_train <- mice(train, m=5, maxit=50, method = "pmm", seed = 500)
mice_train_full <- complete(mice_train, 1)
```

Next order at hand was correctly classifying the data. Some of the columns such as MSSubClass and Id were numeric columns, but these are not continuous values and thus we changed those corresponding columns to characters. After all that was sorted, we ran a function and converted all characters to factors. Now once we did all that, we repeated the same steps for the testing data. The biggest difference between the process between the testing and training set was the values that were missing. We simply followed our thought process from before and used mode imputation if there weren't many of them.

```
mice_train_full[sapply(mice_train_full, is.character)]
<-lapply(mice_train_full[sapply(mice_train_full, is.character)], as.factor)
```

### Checking for multicollinearity

We had a lot of predictors in the dataset, so there is a good chance there will be multicollinearity. We want to eliminate that since it violates our assumption that the predictors are independent, so we used a function that will check our dataset. It will be posted at the end, but what it does is; compare continuous vs. continuous with a correlation matrix, compare continuous vs. categorical using anova tables, and compare categorical vs. categorical with a chi-square test. We then piped this association matrix produced by the function and used the package corrr to see this plot.



We removed all the predictors that had a correlation of over 0.7 since that means they were highly correlated and thus one column held all the information of the other.

After this was completed we checked the levels of the factors for both the test and training set. These levels were not equal so we had to use a new approach. If some levels are significantly associated with SalePrice and there is only 1 observation that is different, we imputed it with the closest possible level. For instance in the Functional category, Sev is most closely related to Maj2 by intuition and therefore we used that. This is quite rudimentary and certainly there are better ways of dealing with this, but we must proceed. If there are too many differing levels and even if the predictor is significantly associated with SalePrice, we must get

rid of it. This is because we cannot predict using a predictor that is out of our sample space, our model has never seen it before and thus it will not predict using it.

After the levels were fixed, we could start building models. We tested random forest using the randomForest package and forward/backward selection using the SignifReg package with the criterion of “AIC”. To test how well they performed, we split the training set into two indices: one was the train set with 70% of the data and the test set with 30%. We could not use the initial testing set because it did not include the SalePrice, so we had to create the split in the training set itself. We then checked the PRESS of all these models.

What we found was the forward selection model had the best PRESS out of the three. Then we checked the diagnostics plot and there appeared to be some heteroscedasticity as well as high leverage points. We removed the high leverage points one at a time, checking to see if they were outliers. Some of the observations did appear to be outliers as there must have been something wrong with them because their prices were not in line with the others. This also helped with heteroscedasticity. That was only our initial assumption however; the prediction power of the model against the actual values went down, so by doing this we created overfitting.

We then went with another method; we created a weight equal to  $1 / \text{StdDev}^2$  and applied it to our models. We then calculated the PRESS for these models against our created testing set and they were lower than the PRESS for the unweighted models. However, the prediction power went down for our best model against the actual training set. The other two models received a benefit from the weight, but our best model ended up being the forward selection with no weights in regards to prediction. Our rank ended up being #1839 on the leaderboards.

### Code ###

All the code can be seen here:

<https://github.com/nickzermeno/House-Price-in-Iowa>