# Ames Housing Price Data: Technical Appendix

In order to keep our presentation focused on the business insights, we have moved technical discussion of the models into this document.

### Data Cleaning and Encoding

#### Filtering and Target Variable Transformation

Unadjusted house sale prices do not follow the normal distribution assumed by many types of statistical models. The diagram below shows a [quantile-quantile plot](https://en.wikipedia.org/wiki/Q%E2%80%93Q_plot) for the sale prices of houses below 4,000 square ft. (We excluded the three datapoints above this area as outliers).

**QQ Plot of House Sale Prices**

A close up of a map

Description automatically generated

The graph clearly shows that the data is not normally-distributed. An optimized [Box-Cox transformation](https://en.wikipedia.org/wiki/Power_transform#Box%E2%80%93Cox_transformation) recommends a λ value of -0.04. This is very close to a log transform, and so for simplicity we apply this. The chart below shows the QQ-plot after taking the log of sale price and indicates a much better fit to a normal distribution.

A close up of a map

Description automatically generated

#### Stationarity of Sale Prices over Time

Given that the period spanned by the data includes the financial crisis of 2007-2008, we checked the data to see how much sale prices changed over the period. The graph below shows monthly average sale prices in USD over the period:

A screenshot of a video game

Description automatically generated

From Jan 2006 to July 2010, the average price of a house sold in Ames dropped by 7.9% However, looking instead at price per square foot, much less of a decline was evident. Price per square foot over the same period dropped by only 1.8%, indicating that fewer large properties sold later in the dataset.

#### Variable Encodings and Transformations

The dataset contains a large number of categorical variables, of which approximately half are ordinal and half are nominal. The most complex of these is the ‘Neighborhood’ variable, which has 28 separate levels. We investigated two separate techniques for encoding these variables. One of these was to follow convention and encode the ordinals using an integer scheme (e.g. Poor, Fair, Good, Excellent maps to 1, 2, 3, 4) and ‘dummify’ the nominal variables into binary classifiers. The other approach investigated was *mean value encoding* in which each level of a categorical variable was replaced by the average value of the log sale price when grouped by that level of the variable. While this approach raises the risk of overfitting, we combined it with penalized regressions to counter it.

Due to the complexity of the Neighborhood variable – and its likely importance in driving price - we handled it separately.

A screenshot of a computer

Description automatically generated

We opted to split this variable into 4 groups, splitting the mean prices at levels below $130k, between this and $170k, between this and$220k, and above this.

#### Missing Values

Three variables had so many missing values we elected to drop them completely. These were: PoolQC, MiscFeature and Alley.

After these, the variable with the largest number of missing values was LotFrontage. We believe it’s unlikely that so many properties in Ames would have no lot frontage, especially when many of these were classed as single family homes. Therefore we concluded this was genuinely missing data and elected to fill values based on the median level for the neighborhood the poperty was located in. We filled 8 missing values in MasVnrArea with zeros after noticing these corresponded to a ‘none’ indicator in MasVnrType. We also defaulted missing data in ‘GarageYrBlt’ to be equal to the house’s YearBuilt, although we suspect that missing data here indicates no garage.

Within the test dataset, a number of other columns had missing values. However, there were no more than 4 missing values in any one of these. We filled these with the median value for the column.

#### Out of Sample Data

We elected to remove a random sample of 20% of the records and keep this as fully out of sample. That is, this data was not used in *any* of the training process , including cross-validation / hyper-parameter selection, and final model choice. Within our in-sample data we implemented a 5-fold cross-validation scheme, effectively creating quasi-out-of-samples sets within it. Only after we selected the final model did we test it on the fully out-of-sample data to validate its accuracy.

### Feature Engineering

We evaluated the features for multi-colinearity. In general, correlations were low except for variables clearly referencing the same fundamental attribute (e.g. Garage area and Garage nuber of cars).

Correlation heat-maps for the continuous and discrete variables are shown below.

A picture containing white, colorful

Description automatically generated

A close up of a logo

Description automatically generated

We elected to scale attributes when used in linear models. Tree-based models and the support vector machine were running using raw data. We computed a price-per-square-foot variable for use in developing recommendations on property upgrades.

### Model Fitting

#### Forward Stepwise Linear Regression

One of the first models we analyzed was a stepwise linear regression starting with just one feature and then adding features until metrics of model improvement indicated we should stop. The two metrics we computed were the R2 of an out-of-sample set selected before the fitting process, the t statistic of each regression parameter and the [Akaike Information Criterion](https://en.wikipedia.org/wiki/Akaike_information_criterion). These metrics indicated that limited value was added beyond the 12th feature.

A close up of a map

Description automatically generated

Significant caution must be applied to the t-statistics and AIC shown above. Forward stepwise regression involves checking a very large quantity of potential feature sets, whereas the t and AIC statistics assume the regression only looked at the number of features actually used. This approach biases R2 and significance levels upwards. That said, we did not see strong signs of overfitting when we switched to out-of-sample data.

#### Penalized Regression Models

We evaluated Lasso and Ridge regressions using 5-fold cross-validation to tune the hyperparameters.

Unfortunately, the strong R2 out of sample resulted in the hyper-parameter tuning recommending values close to a non-penalized multilinear regression. Our best Lasso model retained all of the features, as did Ridge. We decided to report results also for a manually-picked alpha which forced the model to become more parsimonious. An alpha of 100 caused Lasso to restrict itself to 27 variables. While this came at the expense of R2, we believe the simplicity and reduced risk of overfit compensates for this.

#### Tree-Based Models

We evaluated a Random Forests regression and two types of Gradient Boost. The first Gradient Boost model was the sklearn default model, and the second was an experimental application of XGBoost. Further research is needed to fully understand XGBoost to become confident in using it.

Hyperparameter tuning led to the use of 500 estimators in both Random Forests and Gradient Boost. Random Forest optimal tree complexity was comparatively high, at 30 levels. We restricted to a maximum of 30 features, but otherwise left model parameters unchecked. (For example, minimum leaf samples was 1 and we put no upper limit on the sample size). For gradient boosting the default learning rate of 0.1 proved optimal. In contrast to Random Forests, a tree depth of only 2 was optimal.

#### Support Vector Machine

Finally, we tuned a support vector machine. This was the poorest performing model, as well as one of the more computationally intensive. (SVM is O(N2) in the most efficient case). Careful tuning of the C and epsilon parameters was necessary to achieve adequate performance. Final values of 100 (C) and 0.01 (epsilon) were selected. We allowed the model to choose its own gamma (1 / (n\_features \* X.var()).

We also attempted to use a polynomial kernel but found that the model frequently failed to converge. In the end, all useful analysis was performed with the RBF kernel.

#### Summary Results

The results we achieved are summarized in the table below:

A screenshot of a cell phone

Description automatically generated