**Abstract**

Our work concerns a model regression challenge, based on a Kaggle competition that uses data collected for houses sold in Ames, IA. Based on these provided data, we explored various data pre-processing strategies to maximize the effectiveness of a Scikit-learn gradient boosting regressor to create a predictive ensemble for the Kaggle test data set. We designed an algorithm capable of achieving an R-squared value of 0.970 using 10-fold cross-validation on the testing data set, thus presenting a promising approach to creating a reliable predictor based on the provided data features.

**Introduction**

This report encompasses a checkpoint overview for a machine learning gradient boosting regression project as part of a Data Mining course at DePauw University. The project itself is based on a the Kaggle competition, “House Prices: Advanced Regression Techniques,” in which competitors are given a dataset containing properties of recently-sold houses, along with their respective sale prices (for detailed information, see Data Description section). From these data, competitors construct a regressor model using various pre-processing techniques, machine learning regressor algorithms, and algorithm parameterization.

**Data Description**

The provided data set contains a data set with 79 features categories pertaining to different properties of houses (e.g., basement finish, lot size, garage quality, distance to nearest railroad, pool condition, etc.) and one output category for the selling price of the house; each row corresponds to a single house. These features present a mixture of numerical (e.g., date the house was sold) and nominal data (e.g., basement condition).

Our focus in the initial stage centered around “holes” in the data, marked by “NA” values in the data set. In particular, “NA” for certain features has a defined/significant value (e.g., for basement condition, “NA” logically means “no basement”), while in other categories it legitimately signifies a missing data value. Our initial pre-processing focused on discriminating between these different types of “NA” values.

**Experiment: Pre-Processing**

We chose to design our first model using a Scikit-learn gradient boosting regressor, and as such, all features had to be converted to numeric values. This was initially accomplished using the LabelEncoder transform function, which assigns a numeric index to each instance of a unique nominal value. We favored this approach over one-hot encoding (which generates an unnecessary amount of features) and binary encoding (which might foster coincidental relationships between columns in future tests). This approach by itself generated an error when processing “NA” values within otherwise numeric feature groups, and even when successful it treated “NA” as its own value (which was generally incorrect given the nature of the data set outlined in the Data Description section). In response, a function was created to discriminate between these “insignificant NA values” and the “significant NA values.”

Following these pre-processing steps, we developed standardization and normalization functions for the features to decrease bias from large feature values (e.g., year sold). These were implemented in tandem with more accurate nominal-to-numerical conversion functions which focused on nominal features that lent themselves more easily to numerical conversion (e.g., basement quality, exterior condition). These “levels of quality” values were discretized using a normalized set of values equally divided for each level of quality. Following this, corresponding experimental conversions for other nominal values (e.g., masonry veneer type, fence type) according to research yielding the corresponding materials costs (e.g., stone was normalized to a value of 30 compared to cinder block at 3, using Angie’s List). Lastly, broader nominal values were converted to corresponding median sale prices (among all rows containing that value) and then normalized.

**Experiment: Algorithms and Parameterization**

Our initial model was a Scikit-learn gradient boosting regressor, using a random\_state parameter of 1 to minimize bias in model generation. We have not explored any other algorithms or parameterizations currently.

**Experiment: Results**

Our earliest successful test of the gradient boosting regression model (using only the LabelEncoder translation, without normalization/standardization or specific conversion for “levels of quality” values, et al.) yielded a mean R-squared value of approximately 0.887 using 10-fold cross-validation. After including normalization, this accuracy increased to 0.899. Our most performant model currently was a normalized gradient boosting regression model using specific conversion, which yielded an R-squared value of approximately 0.970.

**Experiment: Analysis**

While R-squared values do not directly correlate with accuracy, an R-squared value so close to 1 at this checkpoint stage is very promising for our approach. Additional experimentation will be required to provide greater resolution of normalization/standardization and specific conversion effects between our documented values; furthermore, additional analysis is needed to identify redundant or mostly-redundant features that may bias (or at least muddy) model estimates.

**Conclusion**

While we only have “our foot in the door” at this point, our R-squared value of 0.970 demonstrates a significant step toward a successful and reliable predictor model. We hope in future work to focus on a subset of (at least) elimination of redundant features, continued specific conversion of nominal data, parameterization of the gradient boosting regressor algorithm, and experimentation using other algorithms. We also plan to expand our experimental testing procedures to increase resolution of R-squared data.