House Price Prediction
CIS 572 Machine Learning Term Project - Winter 2017

Manujinda Wathugala
manu@cs.uoregon.edu

Mohsina Zaman
mzaman@cs.uoregon.edu

Himani Chawla
himanic@cs.uoregon.edu

Abstract—The abstract goes here.

1. Introduction

This demo file is intended to serve as a "starter file" for IEEE Computer Society conference papers produced under LaTeX using IEEEtran.cls version 1.8b and later. I wish you the best of success.

mds
August 26, 2015

1.1. Subsection Heading Here

Subsection text here.

1.1.1. Subsubsection Heading Here. Subsubsection text here.

We chose to work with the "House Prices: Advanced Regression Techniques" dataset [**?**] available in Kaggle. The dataset contains 79 features regarding 1460 homes in Ames, Iowa, labeled with the sale price of each property. Sale prices are from year 2006 to 2010. Available features are both nominal and numerical.

There are 44 nominal features and 19 numerical features. We categorized 16 features as ordinal, which include Likert scales, counts, years and months.

By observing the data, we realized that value "NA" has been used to denote two things in the dataset. The 1st usage is to notify a missing value. The other usage is to signal the unavailability of some option such as a garage in a particular home. So, we have to be careful in interpreting the "NA" values in the data by taking into account which column "NA" appears in.

1.1.2. Handling Missing Values. Attributes "MasVnrArea", "LotFrontage", "MasVnrType" and "Electrical" had missing values in the training dataset. For numerical columns we used the column mean and for others we used the column mode as the filler for the missing values.

During the testing, we realized that test data had missing values in other columns not mentioned above. We treated them the same way as we did with the training data. All the fillers were calculated only based on the training data.

2. Methodology

2.1. Feature Selection & Manipulation

2.1.1. Feature Selection. Based on our training data observations, we tried two kinds of feature selections. The methods of the feature selection are as follows:

1) We observed a few attributes had the same value for more than 1400 examples. Since these attributes are almost homogeneous, we surmised that they have very little predictive power on the sale price. Therefore, we decided to drop these attributes in order to reduce the dimensionality of the dataset. In subsequent sections, we refer to this set of reduced features as Reduced Features. [**?**]
2) We also wanted to investigate the predictive power of attributes on the sale price. Therefore, we computed the correlation coefficient of each nominal and ordinal attribute in relation to the sale price. From the computation, we observed there were several attributes with high correlation correlation. As a result, we decided to set a Threshold of 0.1 and use the features with a correlation coefficient greater than the threshold. We will refer to this set of features as Thresholded Features. [**?**]

We built models on each set of features (All attributes, Reduced Features set and Thresholded Feature set) and compared the results.

2.1.2. Feature Manipulation.

- *Normalization:* The dataset has several different types of attributes with values with varying ranges, which might cause these attributes to dominate the prediction decision. Therefore, we decided to use min-max normalization to ensure that all attributes equally contribute to the decision in the absence of weights.
- *Encoding:* As nominal attributes do not display an inherent order, using them in regression without any encoding is meaningless as distance calculation is not possible. Therefore, we decided to use OneHotEncoding to encode the nominal attributes.
- *SalePrice Transformation:* We also observed the SalePrice in the training data to be positively skewed

(Fig. 1), as there are few houses in the higher price ranges. Therefore, we used log transformation on the SalePrice to account for the skew.
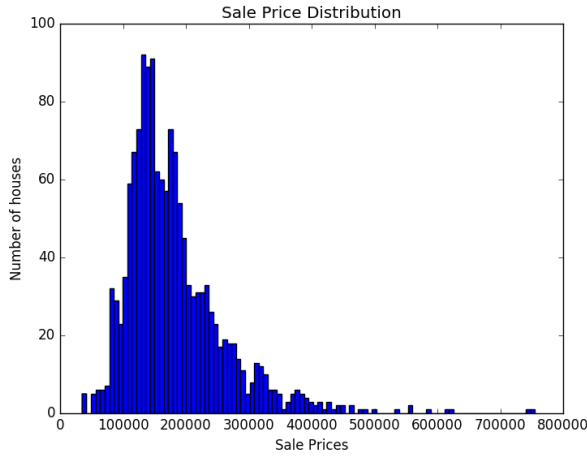

Figure 1. Distribution of sale price

2.2. Models Built

We decided to try and compare several different regression models and a combination of classification and regression models. The models we tried are as follows:

1) *Regressor Models:* We tried several regression models on the different feature sets. The regression models we tried are Lasso, ADA, Nearest Neighbor, Random Forest, Linear and Gradient Boosting regressor.

2) *Classifier and Regressor Combination:* We also implemented a novel model which used a classifier to bin the dataset and a regressor to predict on each bin. For this model also, we tried different classifier and regressor combinations and compared the error for each. We refer to this model as the Custom model.

2.2.1. Model Validation: . As the dataset is small, a reserved validation set greatly reduces our training dataset size. Therefore, we employed standard 5 fold cross validation to train the basic models. However, the Custom model required us to come up with our own variation of 5 fold cross validation, which we elaborate on in the Validation section.

3. Experiments

3.1. Regression Models

As a starting point we tried different regression models to see how good we can do without a lot of effort (Table. 2).

We used Mean Absolute Error (MAE) to compare models since it gives a general idea about how off our predictions are from the actual sale price. However, when using linear regression, lasso and SVR with RBF kernel we had to throw away all the nominal attributes.

For these models we got the least error with Gradient Boosting Regressor with all the features (Fig. 2). The worst error was reported for SVR with RBF kernel, which hints that the problem is more linear. Even with using only the numeric and ordinal featurs, linear regression comparison very close to ensumble models (Fig. 3).
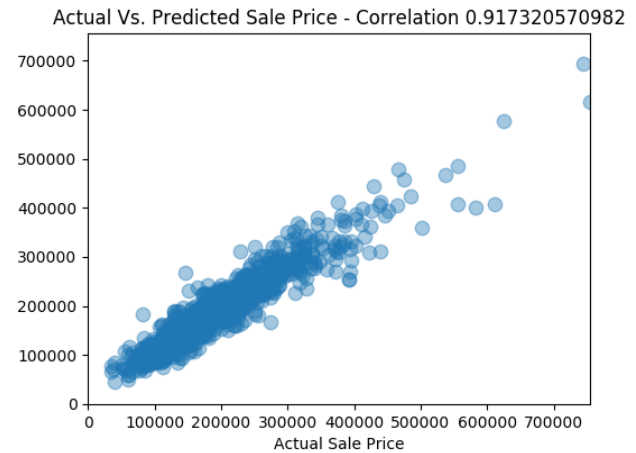

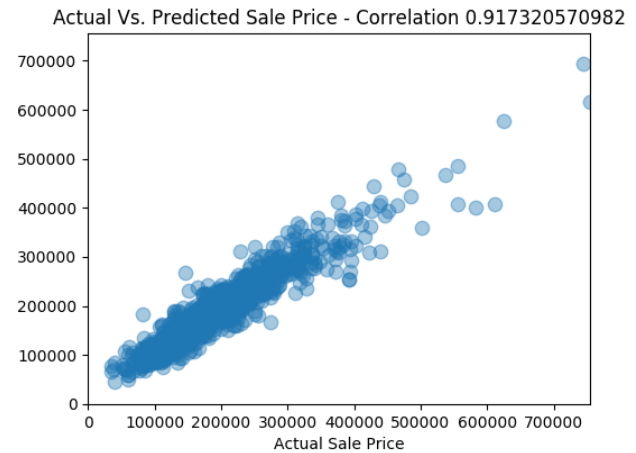Figure 2. Gradient Boosting Regression - Actual vs. Predicted Sale Price

Figure 3. Linear Regression - Actual vs. Predicted Sale Price

To investigate which features are more important, we employed two strategies:

1) We removed all the attributes which had a single value for more than 1400 examples out of the total 1460 training examples ("Reduced Columns" column of Table 2).

2) We computed the correlation coefficient between all the numeric and ordinal with the sale price.

Then we removed all the attributes which had a correlation less than 0.1 ("Correlated + Nominal" column of Table 2).

We trained the same set of regressors with the two sets of reduced features. However, it made the error higher. Gradient Boosting, Random Forest and ADA Boost regressors perform some random feature selection while the model fitting to add diversity to the ensemble of models. This randomness gives them the chance to have better models within the ensemble and that is why they outperform Linear and Lasso regressors.

SVR with RBF kernel performs the worst of all. This gives a hint that the prediction problem at hand is more linear. However, since ensemble models, which do not fit a linear model, outperform normal regression models hint us that trying to fit a single liner model to the entire sale price range is not a good idea.

Moreover, when training Linear and Lasso regressions we had to throw away all the nominal attributes. This gives these models less information to predict the sale price and in a way comparing Linear and Lasso regression models trained without nominal attributes with the ensemble models trained with all the attributes is not a fair comparison.

3.2. Research Questions

Our initial investigation made us curious about two research questions:

1) Can we build a model which could make use of nominal features while using Linear or Lasso regression to predict sale prices?
2) If such a model is feasible, would it outperform or work on par with ensemble models?

Thinking along these research questions lead us toward inventing the our custom model, which pipelines a classifier that classifies each house into a sale price range and a sequence of Linear or Lasso regressors that try to predict sale price within that bin. Custom Model subsection dives deep into our custom model and the experience we had with it.

3.3. Custom Model

Standard regression models work better with numerical attributes. Although, classification models such as decision trees are initially designed to work with nominal features, the algorithms are extended to work with numeric data too. Since, the dataset has both nominal, categorical and ordinal attributes, we thought to experiment with a novel idea which matches groups of each attributes of each type with the models that work well with such attribute types.

3.3.1. Basic idea: . The model combines a classifier and a regressor. The classifier classifies each example into a sale price range, which is represented by a bin. Then a regression model is used to predict the actual sale price of the example with finer granularity within the bin by employing a regression model trained for that bin.

3.3.2. Hypothesis: . At the regression end, more the bins, smaller the variation among sale prices that falls within that bin. This leads to a better sale price prediction. Further, since the distribution of the sale prices are skewed, trying to fit a single regression model to the full range might not work well. In this light, fitting several regression model to different ranges of sale prices would provide a piecewise approximation to the sale price prediction which could outperform the single model. However, on the classification side, more the bins, more classes to predict leading to higher classification error.

There should be some number of bins that balances the bin classification error and the regression prediction error, which minimizes the combined model error. This makes the number of bins a hyper parameter of this novel custom model.

3.3.3. Training: . Based on our hypothesis, we expect the validation error to initially reduce and then start growing when we increase the number of bins. Thus, to find the best number of bins, we trained multiple versions of our model increasing the number of bins from 1 to a maximum number. While this training is going on, we keep track of the validation error and watch for the moment it starts going up. We keep note of the number of bins that gave the minimum validation error thus far. Note here that we do not pick the number of bins that gives the minimum error for the whole training run, but the number of bins we observ the error start growing for the first time. Choosing the number of bins that gives minimum for the entire run quite possibly be over-fitting the data.

Partitioning the data to perform 5 fold cross validation while training the custom model should be done with extra care. On one hand, if we use standard cross validation to train both the classifier and the array of regressors at once, we run the risk of some bins being empty or having very little training (or validation) samples during training (or validation) due to the effect of random sampling. This greatly affect the training (or validation) of the regression models. On the other hand, if we train classifier and regression models separately using 5 fold classification, we have to use the validation set of the classifier as the validation set for the full model. This is problematic since the regression models could have seen some of these held out data because the held out validation sets for classification and regression models are computed independent of each other.

As the first step to reduce the danger of training (or validation) data sparsity for some regression models, we decided to use equal frequency binning to assign a bin label for each example. This makes sure that each regression model initially have a similar number of examples to train on. Making sure that we have an uncontaminated validation set throughout the training process took a little bit more work (Algorithm 1).

The training process provides the ideal number of bins and the validation error of the custom model (the red curve in Fig. 5). We separately keep track of the classification error (Fig. 4) and regression error given correct bin labels

TABLE 1. VALIDATION ERROR - DIFFERENT MODELS VS. DIFFERENT FEATURE SETS

	All Attributes	Correlated + Nominal	Reduced Columns
GradientBoostingRegressor	16118.18	16669.75	16256.96
RandomForestRegressor	18906.08	19091.70	18961.65
AdaBoostRegressor	22904.15	22974.78	22861.48

TABLE 2. VALIDATION ERROR - DIFFERENT MODELS VS. DIFFERENT NOMINAL & ORDINAL FEATURE SETS

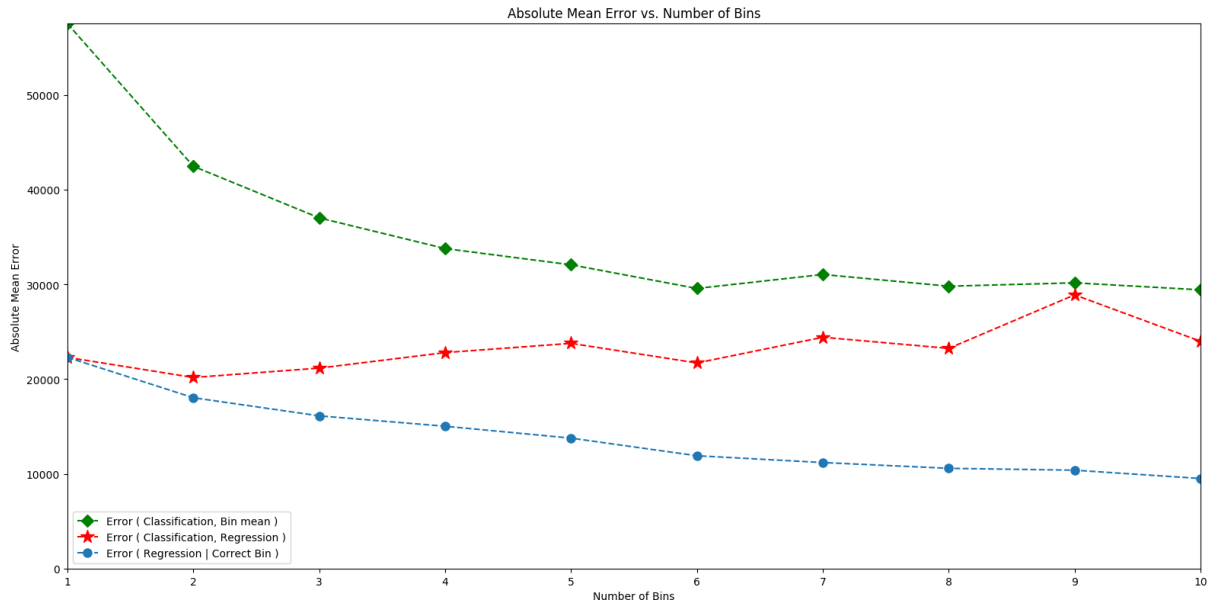|	All Attributes	Correlated+Nominal	Reduced,Columns
LinearRegression	19430.39	20851.57	19963.10
Lasso	30170.06	30138.92	30139.63
SVR,(kernel = 'rbf')	55954.24	55954.39	55954.31

Figure 5. Custom Model Training - Regression Errors vs. Number of Bins

(the blue curve in Fig. 5) to peek inside the custom model to see what is working write and what is not. The blue curve provides a lower bound to the model. Options at hand to try to get a error below the blue curve are finding a better regression model, feature engineering or getting better features.

We further keep track of the error if we predict bin average as the prediction (green curve in Fig. 5) . This provides an upper bound for the error.

We used the bin average as a prediction to house prices in all the houses that fall into a single bin to gauge how well the model is working. With bin average as the prediction we kept track of custom model error and the error given the correct bin. These errors provide an upper bound for the model.

3.3.4. Custom Model - Expriments: . Our initial choice for the classifer-regressor pair for the custom model is decision trees and and linear regression (Fig. 6).

4. Error Analysis:

As we can see from the graph in Fig. 4, the difference between the blue and red curve denote the area our model can improve on, for better prediction. As the complex model consists of the two stages: the classification and regression stage, we can attempt to improve either or both of them. However, from our previous experiments, we know the regression models perform quite well on the numeric and ordinal attributes, which indicates the classification stage has room for improvement. The further proof of this is shown in Fig. 6, which shows decision tree classifier and linear regression to have a correlation of 0.885, with an optimum bin number of 2 bins. This correlation is lower than the correlation of 0.917 when using simple linear regression on only linear and ordinal attributes. We contribute the reduction in performance to incorrect classification of houses into bins, or incorrect binning scheme, which are both part of the classification stage. Therefore, we tried two methods to improve the classification stage.

Algorithm 1 Algorithm for training custom model

Input: folds, maxbins
Output: least_error, best_no_of_bins
1: least_error = maxint
2: best_no_of_bins = 0
3: **for** $bins = 1$ to $maxbins$ **do**
4:    Assign bin labels to examples
5:    Randomly partition examples in each bin to folds (let partition f of bin b as F_{bf})
6:    model_fold = 0
7:    **for** $f = 1$ to $folds$ **do**
8:       validation_set = $\sum_{b=1}^{bins} F_{bf}$
9:       training_set = $\sum_{b=1, i \neq f}^{bins} F_{bi}$
10:      Train $classifier$ on training_set
11:      Validate $classifier$ on validation_set
12:      Assign predicted bin labels to validation_set examples
13:      Divide validation_set based on bin labels (let validation set for bin b be V_b)
14:      error_fold = 0
15:      **for** $b = 1$ to $bins$ **do**
16:         regressor_training_set = $\sum_{i \neq f} F_{bi}$
17:         Train $regressor_b$ on regressor_training_set
18:         Validate $regressor_b$ on F_{bf}
19:         model_error = Validate custom model on V_b
20:         error_fold ← error_fold + model_error
21:      **end for**
22:    **end for**
23:    model_error ← $\frac{model\_error}{folds}$
24:    **if** model_error $\leq$ least_error **then**
25:      least_error = model_error
26:      best_no_of_bins = bins
27:    **else**
28:      **return** least_error, best_no_of_bins
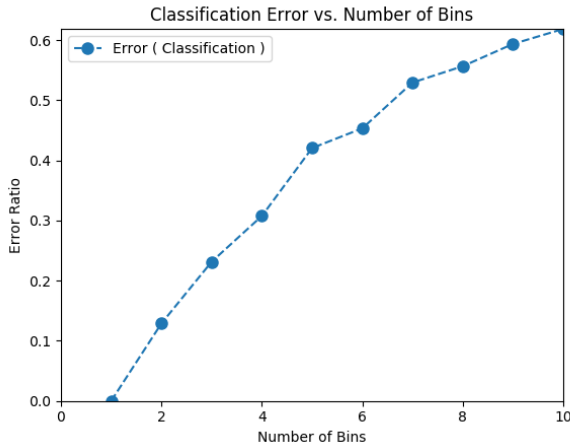29:   **end if**
30: **end for**

Figure 4. Custom Model Training - Classification Errors vs. Number of Bins
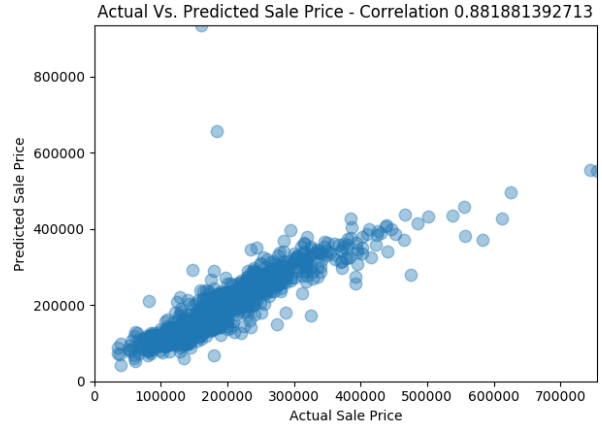

Figure 6. Custom Model Decision Trees + Linear Regression - Actual vs. Predicted Sale Price

- *Different Classification Algorithm:* Given the binning scheme, it is possible the existing clssification algorithm is not able to learn the association between attributes of houses and the assigned bin labels well. Therefore, a different classifier might be able to learn the binning better, resulting in coherent bins, which in turn improves the accuracy of our regression stage.
- *Different Binning Scheme:* It is also possible that the existing binning scheme does not give rise to a learnable pattern between feature vectors and assigned bin lables. This renders any classifier hopeless. Therefore, a different binning scheme, which bins homogeneous houses into a single bin, can also improve performance.

5. Improved Custom Model

We tried different models for each of the possible options mentioned above.

5.1. Different Classification Algorithm

We tried several different classification algorithms. We also tried different combination of classifier and regression models. Since Gradient Boosting and Random Forest regressors worked so well in the regression models, we decided to attempt gradient boosting and random forest classification and regression models together. Table 3 shows the correlation for each of the models. As we can see, Gradient Boosting Classifier and Regressor model performs the best with a correlation of 0.921, with an optimum bin number of 2 bins, which is close to the best model of using simple gradient boosting regressor model, which had a correlation of 0.947, and also an improvement from our original custom model.

TABLE 3. CUSTOM MODEL - ERRORS AND CORRELATIONS

	Mean Absolute Error	Correlation	Optimal Bins
Decision Tree & Linear Regression	20,283	0.881	2
Random Forest Classifier & Regression	19,279	0.910	1
Gradient Boosting Classifier & Regression	17,957	0.921	2
K-means, Decision Tree & Linear Regression	22,393	0.867	1
Gaussian Mixture Model, Decision Tree & Linear Regression	22,429	0.867	1
K-means, Gradient Boosting Classifier & Regression	18,201	0.919	2
Gaussian Mixture Model, Gradient Boosting Classifier & Regression	17,853	0.921	2

5.2. Different Binning Schemes

Given that we are using equal frequency bins in the original model, it is possible that houses which are significantly different from each other could be put in the same bin, which could be the reason that decreases the accuracy of our classification models. This is especially true for the last bin, as there are very few houses with high sale prices, these houses might be put in the same bin as some significantly lower priced houses. Therefore we hypethesize, a scheme which forms more coherent bins where the differences in the houses that fall within one bin are not as signifiant as equal frequency binning can improve binning. Following this logic, we decided to try K-means clustering and Gaussian Mixture Models to do binning, which are then fed to the classifier. We attempted four different combinations, which are

- K-means clustering with Decision Tree Classifier and Linear Regressor
- Gaussian Mixture Model with Decision Tree Classifier and Linear Regressor
- K-means Clustering with Gradient Boosting Classifier and Regressor
- Gaussian Mixture Model with Gradient Boosting Classifier and Regressor

We decided to attempt gradient boosting as well, as it is the best performing classfier and regressor thus far. Table 3 shows the performances of these four models. Both k-means and gaussian mixture model with decision tree classifier and linear regression achieve a correlation of 0.867, which is an improvement from our original model. K-means and gaussian mixture model with gradient boosting classifier and regressor achieve a correlation of 0.921, which is close to our best model.

why k-means and gmm get similar results.??

From our results, we observe different classification algorithms and binning schemes both improve the performance of the original custom model with Decision Trees and Linear Regression, and the use of ensembles significantly improve the accuracy of the models.

However, accroding to table 3, the optimal number of bins for the the best performing models is one with Gradient Boosting Classifier being the only exception. This is also due to the fact that the Gradient Boosting Classifier does not work with a single class. Further, comparing the results for using just the Gradient Boosting Regressor and the custom model with Gradient Boosting Classifier and Regressor, just using the Gradient Boosting Regressor works better.

Custom model with a single bin means that the classifire does not add any improvement and it is equivalent to a simple model with only the regressor. Therefor, with this observation what we can conclude is that the custom model we have thus far fails to improve beyond the best results we get with a simple model.

6. Conclusion

We started our investigation on the Kaggle housing price data by carrying out some feature manipulation and selecting and running simple regression models. Here we observed, ensemble models, especially gradient boosting worked best when using all attributes and linear regressor performed close to it using only numeric and ordinal attributes, which gave way to two research question. We were able to answer the first, by building a custom model using a combination of a classifier and regression model. We also improved the custom model by introducing the use of different classification algorithms and binning schemes.

The performance of our improved model was close to our best achieved performance from all models, which answered our second research question. We believe more intuitive feature selection and other classification and binning approaches will be able to improve the performance of our custom model.

Acknowledgments

The authors would like to thank...

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.