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 focus of this study is to find the best way to predict the sale price of a house based on 79 features that describe a house. We tested several simple regression models. The major contribution of the study is researching with a novel idea to combine a classification model with a regression model and observing whether the performance improves. Although we were successful in building a model capable of combining a classifier and a regressor into a single model, the final model did not outperform the best simple models we tested.**

1. Introduction

This is a regression project which aims at predicting the sale price of a house given an array of attributes about a house. The dataset we chose to work with is the "House Prices: Advanced Regression Techniques" dataset available in Kaggle.

1.1. Dataset

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 had to practice caution in interpreting the "NA" values in the data by taking into account which column "NA" appears in.

1.1.1. 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 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 low 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.

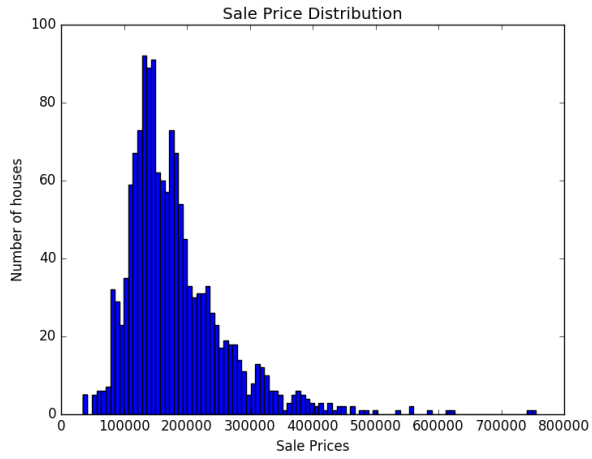

Figure 1. Distribution of sale price

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, ADABoost, 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 provides a general idea about how deviated our predictions are from the actual sale price. However, when using linear regression, lasso and SVR with RBF kernel we were unable to use the nominal attributes.

For these models we obtained the least error with Gradient Boosting Regressor when using all the features (Fig. 2). The worst error was reported for SVR with RBF kernel, which suggests the problem is more linear. Even while using only the numeric and ordinal features, the performance of linear regression is comparable to the ensemble models (Fig. 3).

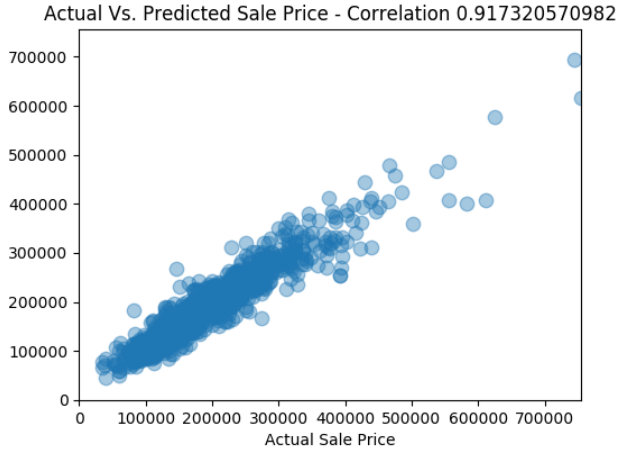

Figure 2. Gradient Boosting Regression - Actual vs. Predicted Sale Price

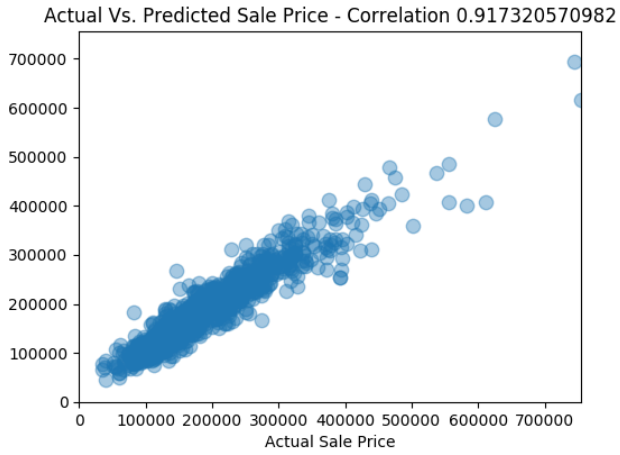

Figure 3. Linear Regression - Actual vs. Predicted Sale Price

We trained the same set of regressors with the two sets of reduced features. However, it caused the error to increase. Gradient Boosting, Random Forest and ADA Boost regressors perform some random feature selection while fitting the model to add diversity to the ensemble of models. This randomness gives them the chance to have better models

within the ensemble and this is why they outperform Linear and Lasso regressors.

Since we observed that SVR with RBF kernel performed the worst, which suggested that the prediction problem at hand is more linear. Further, since ensemble models, which do not fit a linear model, outperform normal regression models suggests that attempting to fit a single linear model to the entire sale price range is not a good idea.

Moreover, since to train Linear and Lasso regressions, we only used a subset of available features (numeric and ordinal), comparing their predictions with the ensemble models which use all of the available features is unfair.

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 led us towards inventing 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 the bins. 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 nominal, categorical and ordinal attributes, we attempted experimenting 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, the number of bins is inversely proportional to the variation among sale prices that fall within one bin (as number of examples within a bin also decreases), which leads to a better sale price prediction. Furthermore, 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, the number of bins is proportional to the number of classes to predict, which leads to higher classification error.

Therefore, we surmise, there should be an optimum number of bins which balances the bin classification error and the regression prediction error, and consequently 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 to grow as the number of bins increase further. Thus, to find the optimum 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 which results in the minimum error for the whole training run, but the first bin after which the error starts growing. Choosing the number of bins that gives minimum error for the entire run quite possibly results due to 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 few training (or validation) samples during training (or validation) due to the effect of random sampling. This greatly affects 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 errorprone 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 risk 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 more effort (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 kept track of the classification error (Fig. 4) and regression error, given correct bin labels (the blue curve in Fig. 5) to peek inside the custom model to see what is working right and what is not. The blue curve provides a lower bound to the model. Options to try to get a error below the blue curve are finding a better regression model, feature engineering or obtaining a better feature set.

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.

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

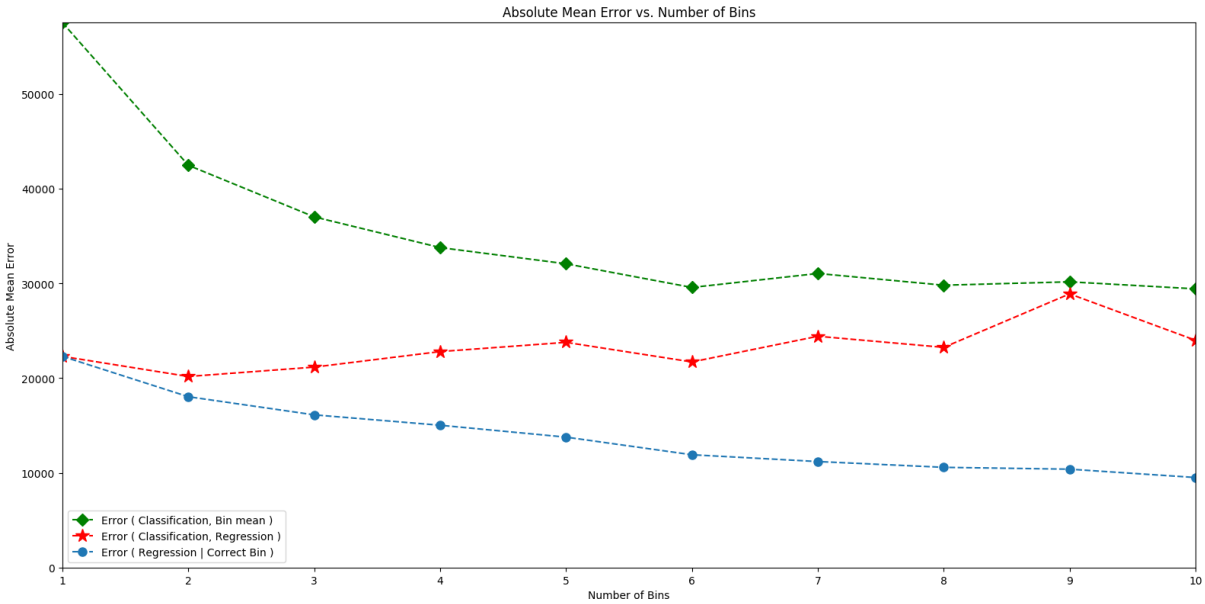

Figure 5. Custom Model Training - Regression Errors vs. Number of Bins

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

4. Error Analysis:

As we can see from the graph in Fig. 5, the difference between the blue and red curves denotes 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 0.881, 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.

- *Different Classification Algorithm:* Given the binning scheme, it is possible the existing classification 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.

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

Algorithm 1 Algorithm for training custom model

Input: folds, maxbins
Output: least_error, best_no_of_bins

least_error = maxint
best_no_of_bins = 0
for $bins = 1$ to $maxbins$ **do**
  Assign bin labels to examples
  Randomly partition examples in each bin to folds (let partition f of bin b as F_{bf})
  model_fold = 0
  for $f = 1$ to $folds$ **do**
    validation_set = $\sum_{b=1}^{bins} F_{bf}$
    training_set = $\sum_{b=1,i\neq f}^{bins} F_{bi}$
    Train $classifier$ on training_set
    Validate $classifier$ on validation_set
    Assign predicted bin labels to validation_set examples
    Divide validation_set based on bin labels (let validation set for bin b be V_b)
    error_fold = 0
    for $b = 1$ to $bins$ **do**
      regressor_training_set = $\sum_{i\neq f} F_{bi}$
      Train $regressor_b$ on regressor_training_set
      Validate $regressor_b$ on F_{bf}
      model_error = Validate custom model on V_b
      error_fold ← error_fold + model_error
    end for
  end for
  model_error ← $\frac{model\_error}{folds}$
  if model_error ≤ least_error **then**
    least_error = model_error
    best_no_of_bins = bins
  else
    return least_error, best_no_of_bins
  end if
end for

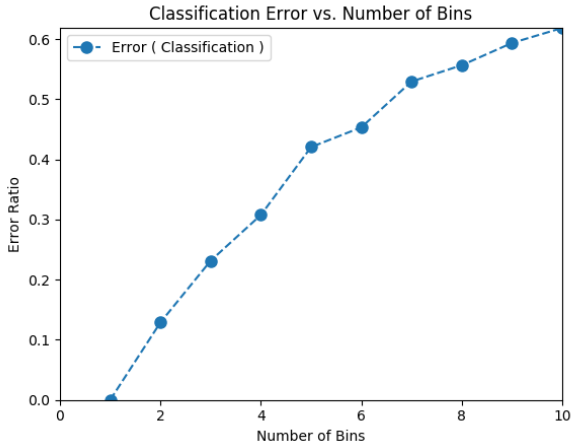

Figure 4. Custom Model Training - Classification Errors vs. Number of Bins

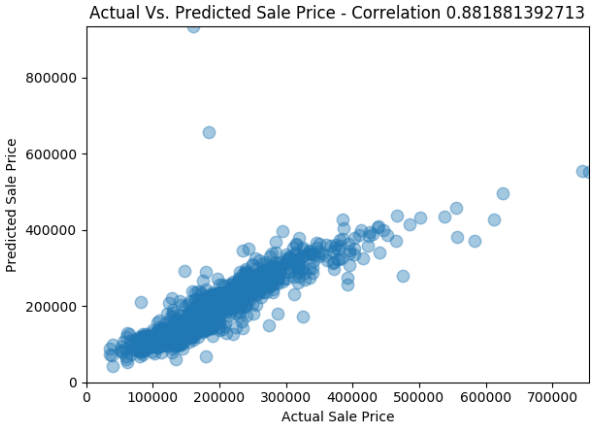

Figure 6. Custom Model Decision Trees + Linear Regression - Actual vs. Predicted Sale Price

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 combinations of classifier and regression models. Since Gradient Boosting and Random Forest regressors worked 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 perform 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.

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.

6. Discussion

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, according to the 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. Furthermore, comparing the results of the simple model which uses just the Gradient Boosting Regressor with the custom model which uses Gradient Boosting Classifier and Regressor, we observed the simple model performs better.

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

Regressor versions of the ensemble models are built upon the classifier version of the respective models. This could most probably be the reason why when ensemble model classifier and regressor pairs plugged into custom model not being able to outperform directly using the regressor version of the ensemble models. This effect is visible in the scatter plot in Figure **??** having two visible clusters, which are for the two bins that the respective model used to make the predictions on.

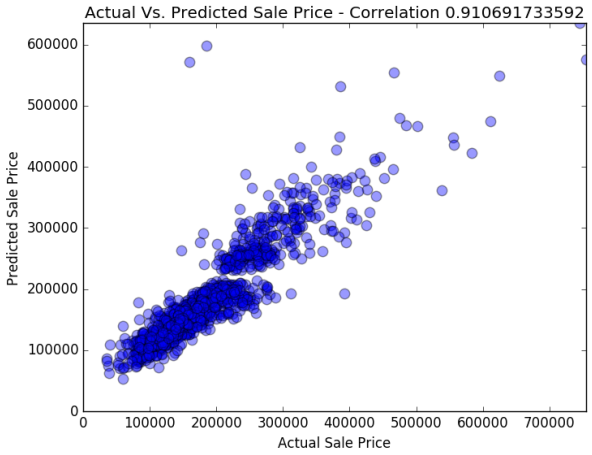

Figure 7. Effect of Binning on Regression Prediction

Moreover, in the custom model, a misclassfication in the bin predictional level is very costly since the final price will be predicted based on a regression model totally irrelevent to that house. The further the predicted bin is from the actual bin, the more the error. With the direct regression models, such effect could be mild since we estimate the house price based on a model trained on all the houses.

7. 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 research question 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 custom model was able to slightly improve the performance of Linear and Lasso regression. However, it failed to add any improvement to ensumble models.

More feature engineering with the aid of domain knowledge, further investigation into defining better binning and bin classification algorithms are the future directions that we can take to improve the prediction accuracy of this problem.