# Module 1: Housing Prices Report

Kevin Hua, Tyler Oh, Linden Gueck

Oct 20, 2023

STAT 440 D100

## ***Data***

The data we received consists of the training (Xtr.csv and Ytr.csv) and testing set (Xte.csv). The training set has 199,246 items, while the testing set contains 796,398 items. Each item consists of 8 variables. The "ID" is a unique identification number for each data point. Our response variable in Ytr is "Sale_Amount" which is a numerical variable. Our other numerical variables were "Assessed_Value", which has a 50% missing rate, and "List_Year", spanning from 1999 to 2020. The dataset also includes the categorical variables "Town" with 169 unique town names, "Address", which contains a few missing observations, and "Property_Type" and "Residential_Type", both showing a 50% rate of missing observations. Finally, "Date_Recorded", represented in datetime format, is the date of real estate sales.

Both the training and testing datasets had a problem in terms of missing values, particularly "Assessed_Value", "Residential_Type", "Property_Type" and issues related to the formatting and consistency of the "Address" Column. After some discussion, we first resolved inconsistencies between the "Residential_Type" and "Property_Type" columns. Property_Type values like "Single Family", "Two Family", etc., were converted to "Residential" since they all represent residential property types. "Vacant Land" in Property_Type was also set as "Vacant Land" in Residential_Type, and similar conversions were done for "Industrial" and "Commercial". In addition, we performed common data cleaning techniques for the Address column by removing special characters, normalizing the formatting, and resolving redundancy or variations. The remaining missing "Address" values were imputed based on the mode within each town, and missing "Assessed_Value" observations were filled using the median value within the same "Address" group. Any remaining missing "Assessed_Value" entries were replaced with zeros since no information is available. We later decided to use the logarithm of Assessed_Value as standardization.

Figure 1 displays the log of the median sale amount for each day represented in our data set. It clearly shows an increasing trend from our first point until about halfway between 2005 and 2010, where our prices drop drastically. Shortly before our 2010 mark, prices level out, and remain that way until shortly before 2020, when prices start to sharply rise again. In addition, we can observe a faint sinusoidal pattern, perhaps indicating less drastic fluctuations of house prices by year. Figure 2 shows the pairwise plots of the log of the median for each numerical value by date recorded. In Figure 2 all missing Assessed_Value entries were replaced by the median Assessed Value. Unsurprisingly, List_Year has a very clear correlation to Date_Recorded, though it is interesting to note the few outliers. The only other obvious strong correlation is between Assessed_Value and Sale_Amount, especially if you ignore the line in the middle from the NA conversion.

For our features, we created new features named "Ratio" and "F1B". To calculate the "Ratio", the Sale_Amount was divided by the Assessed_Value, capturing the relationship between the sale price and property's assessed value. Also, we computed KNNImputer to handle missing Ratio due to missing Assessed_Value. Moreover, we created F1B by the mean Sale_Amount for each unique town, effectively capturing market trends by town.

## ***Methods***

For our methods, we fitted linear regression, random forest, lasso regression, decision tree and gradient boosting. Lasso regression is similar to linear regression but it adds a penalty term that shrinks coefficients. Decision tree is a type of supervised learning that predicts the target value by making a series of decisions and the value is determined by the node of the leaf. Random forest is similar to decision trees but uses multiple trees so it is  Finally, gradient boosting is a robust, complex version of decision trees. The reason we chose these models over simple linear regression is that these models are well adjusted to large datasets.  In addition, these models improve regularization: they all help to prevent overfitting.  We optimized model parameters through cross-validation, employing techniques such as grid search and random search for the best hyperparameters.  As a result, we decided to use learning_rate and colsample_bytree as parameters for the XGBoost model.

In regards to feature engineering, we started by creating F1B which is the mean assessed value for each town and imputed it.  We next started considering various features and their impact on model performance. We experimented with a range of feature combinations, recording their RMSEs. One of the initial considerations was which variables we would like to keep for our final model. Hence, we did a stepwise approach, adding or subtracting one variable at a time from our model, allowing us to evaluate the influence of each feature. Furthermore, we tried to standardize 'Assessed_Value' since it had potential outliers and zeros in it by using Min-Max Scaling, normalization to the [0,1] range, and taking the logarithm, eventually settling on using the logarithm in our final model.  We also attempted to create new features by taking the interaction between columns.  However, they did not help to predict the complex relationships within the data, so were ultimately scrapped.

For iterating feature engineering, we considered interaction terms with Property_Type and Residential_Type as well as adding features.  However, when we submitted our models on Kaggle in addition to performing cross validation, our model with the interaction terms had high RMSE on the leaderboard, and when we used 10-fold cross validation, the features we used gave us the best prediction (ratio and assessed values) performed the best with the lowest RMSE. After creating each model, we compared its RMSE and interpreted the feature importance plot to decide whether to keep or remove variables (Figure 4). From there, we tested adding or removing one variable at a time, repeating our comparisons each time.

## ***Results***

We performed 10-fold cross validation to evaluate RMSE over five different methods. As shown in Figure 3, the ensemble methods, such as XGBoost and RandomForest, perform well in capturing the complicated patterns in the data. On the other hand, decision trees rank as the least effective, with the highest median and mean RMSE. LASSO and Linear Regression show similar RMSE values, which also failed to capture complex relationships in our dataset. XGBoost worked exceptionally well, likely due to its ability to minimize overfitting and to capture complex patterns in the dataset.

To improve our results, we could spend more effort to deal with missing Address values. We could also find a way to better utilize our Residential_Type and Property_Type columns in an attempt to capture any relation between these columns and Sale_Price. We could also find a better method to estimate our missing Assessed_Value numbers, since they have such a strong correlation to our Sale_Value (as shown in Figure 2). For instance, we could estimate missing Assessed_Values to be equal to the median by Residential_Type and/or Property_Type. We could also use List_Year or Date_Recorded to help estimate our missing Assessed Values.

There are more machine learning techniques that we could implement for our prediction such as neural networks, which perform exceedingly well when dealing with datasets with such high volume and high dimensionality. While we briefly attempted to use this powerful tool, we were overcome by computer errors and were forced to abandon the attempt. Finally, the large dataset size limited our capacity for extensive hyperparameter tuning. Since each model took so long to calculate, we had to be more stingy with our model tuning than we would have otherwise preferred.

# ***Appendix***

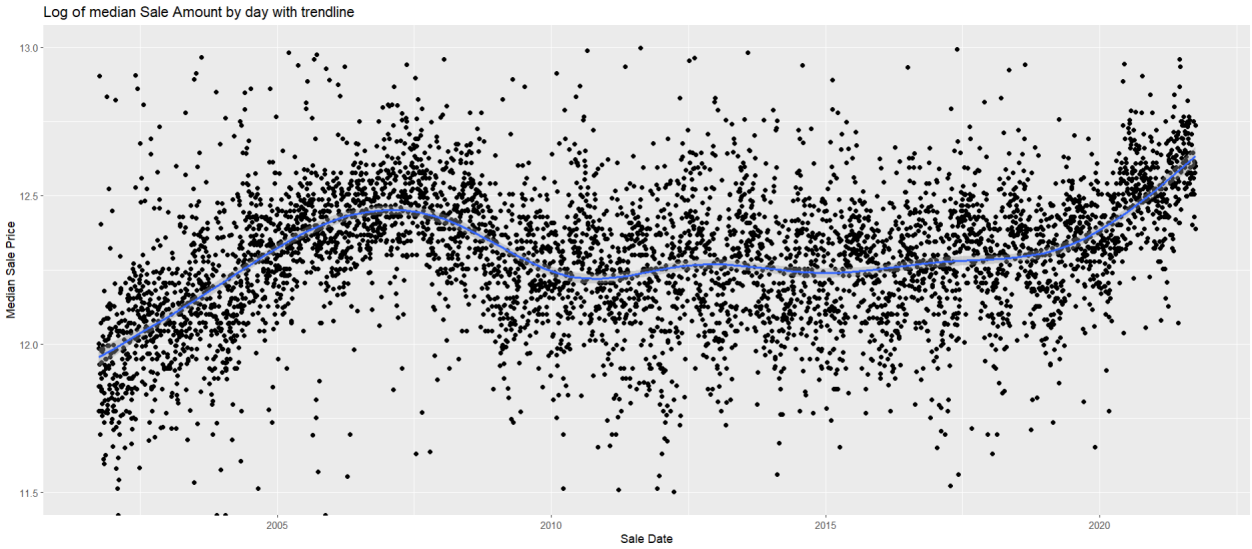


Figure 1: Plot of sale prices/assessed value using ggplot2's built-in trendline

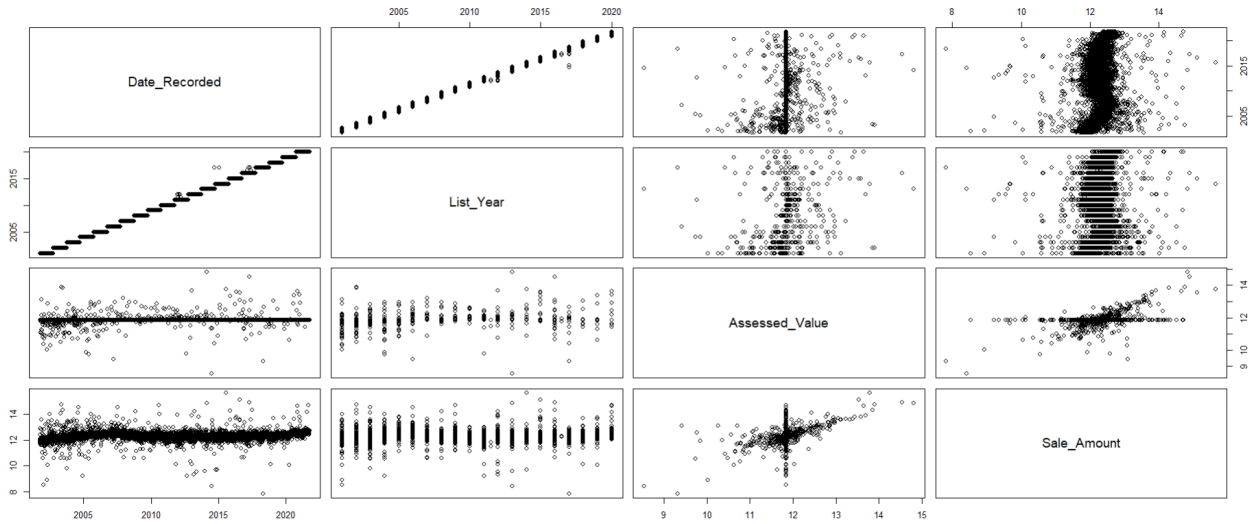


Figure 2: Plots of median log values by date

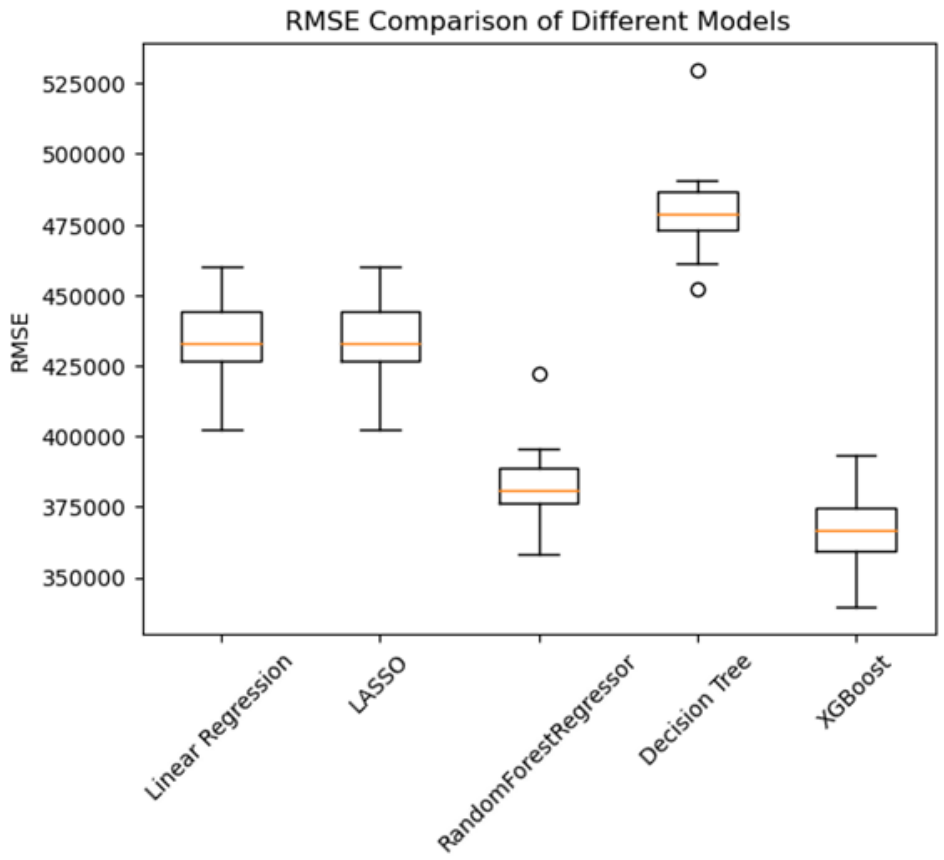


Figure 3: RMSE of 5 methods

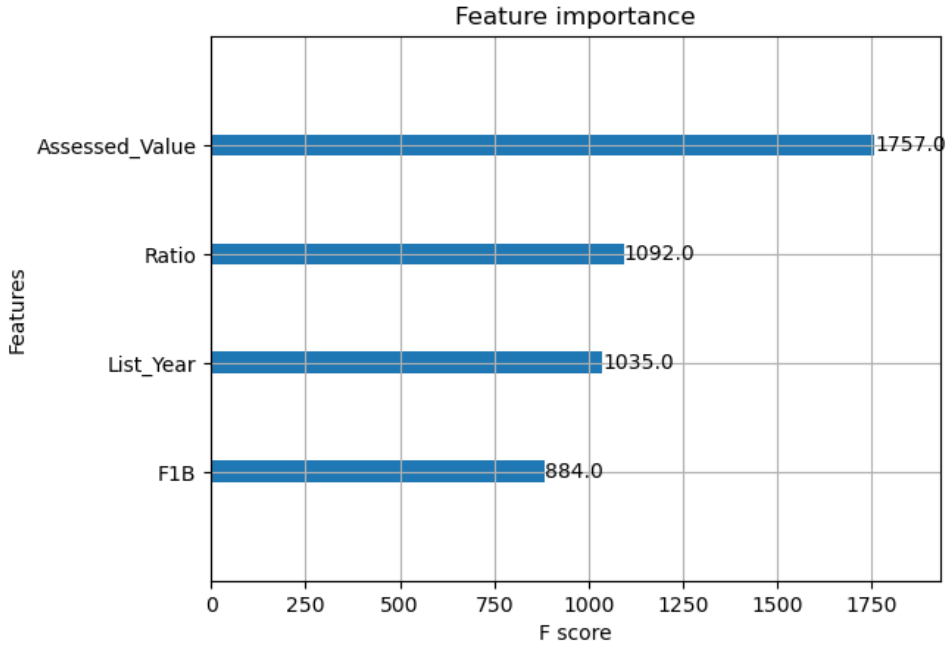


Figure 4: Final model's Feature Importance plot