UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Machine Learning Project

Property Value Assessment in Buffalo, New York

Mateo Jácome
Kathryn Weissman

26/05/2022
Professors Mario Martín Muñoz & Raquel Pérez Arnal
MDS - Semester 2
FIB - UPC

Table of Contents

1. Introduction

Property taxes are charged annually to property owners in the city of Buffalo, New York, and the tax is based on the assessed value of their property [1]. We expect properties with similar characteristics to be valued similarly, and we experiment with different machine learning techniques and regression models in order to accurately predict a property's value given a set of features.

The data we used is available at Open Data Buffalo [2]. The raw dataset contains 93,862 rows and 51 columns. Each row is a unique property, which could be a land lot only, or a property with buildings on it. Columns include a variety of numeric and categorical variables related to the property's features, location, and ownership. As will be explained below, we used a subset of the original dataset, which included properties from the two largest categories of "Property Class", a variable indicating the type of property.

The experiments we have performed and will present in this document include four types of linear regression: Ordinary Least Squares, Lasso, Ridge, and Huber, and also some non-linear models: Decision Tree and Random Forest models. The code used to produce all of our models was designed to be as reproducible as possible and is available for inspection on our Github repository [3].

2. Previous Related Work

There has been a lot of interest in using machine learning to predict real estate prices from both the private sector and publicly funded academic research institutions. In 2017, the US-based company Zillow launched a data science competition on the Kaggle platform with the goal of building a new home-valuation algorithm to predict the sale price [4].

The experiments by García-Magariño et al. used machine learning to estimate missing home prices for an agent-based simulation of real estate transactions in three neighborhoods of Teruel, Spain [5]. They compared results of four different regression models, including linear regression, support vector regression (SVR), k-nearest neighbors (KNN) and multilayer perceptron (MLP) combined with three different dimensionality reduction methods including nonnegative matrix factorization (NMF), recursive feature elimination (RFE), and feature selection with a variance threshold. They used a tenfold cross-validation with a sample size of 89 houses. Their results indicate that SVR was better than other regression models, and that RFE was the best dimensionality reduction technique. RFE selects features based on regression performance.

3. Data Exploration and Preprocessing

Prior to exploring the data, we partitioned the data into the Train, Validation, and Test sets. We only analyzed the data in the Train set so that any decisions used during feature selection, modeling and training would not contain bias based on the data used for validation and testing. We used sklearn's train_test_split utility, ensuring that the rows were shuffled before performing the split in order to prevent any patterns in the original ordering of the data to introduce biases in our split [6].

3.1. Structural Exploration and Preprocessing

After splitting the dataset into the different partitions, we started exploring the distributions and relationships of our variables in the training dataset. Out of the original 51 columns, we quickly identified a set of at least 9 columns that contained a very high number of missing values (>90%). We removed those variables right away, given their low quality and little amount of valuable information.

We then identified some columns with redundant information: columns such as "City" or "State" had only one value, since all of the properties in the dataset are located in Buffalo, New York. We also found that some of our columns are encoded. When we found this and there was information about the encoding available, we decoded the column into its human-readable version and then removed the encoded column (given that its information is redundant).

For variables that contain textual information, such as owner names, street names, mail addresses, etc, we attempted to extract potentially valuable information from these variables by performing some feature engineering. We identified some patterns that were slightly associated with our target variable, and created categorical variables to reflect that information. We ultimately dropped the original textual variables, such as "Mail1", "Street", or "Street Number".

During data exploration, we realized that our dataset presented some problems regarding the structure of the data: some properties had a building on them and others don't, some rows systematically had a higher number of values missing. This is because a vacant land lot can't possibly have an assigned number of bedrooms, bathrooms, or a living area. This represented a structural schism that separated our dataset into two virtually separable datasets: one for properties with buildings, containing a whole set of variables about building features, and one for properties without buildings, containing a more limited set of variables. At this point, we decided to restrict our project to the more rich part of the dataset: properties with buildings, and we decided to specifically focus on properties classified as one- and two-family dwellings: the two most abundant property classes in the dataset, representing 40% and 28% of all the rows in the dataset, respectively.

3.2. Statistical Exploration and Outlier Removal

To explore the statistics of the different variables in our dataset, we observed the minimum and maximum values, together with the quartiles and the mean and median of the variables. We identified that some of the variables had clear outliers, for which we decided to use a basic strategy for outlier detection, based on 3-fold interquartile range (IQR). We visualized

the histograms of the different variables with their IQRs, and decided on the cutoffs for those variables with outliers. Some of the visualizations can be consulted in figures S1 to S4.

During the statistical exploration we found that some of our variables had encoded missing values, such as dates like “09/09/9999”, or areas of 0 square feet. We identified these artificial outliers that were in fact encoded missing values, and replaced them with NA values. Besides the encoded missing values, there were two types of outlier removal to be done; for the target variable, where rows with an outlier would be completely removed from the dataset; and for the predictor variables, where only the outlier value would be removed.

Regarding the target variable, we found that the IQR strategy was very restrictive, with a cutoff of approximately $200.000, leaving out almost 5% of our rows. We considered this too limiting, and we decided to raise the cutoff to $400.000 based on the number of removed rows. We wanted the removed rows to be well under the 1% of the total rows.

Regarding the predictor variables, we decided to leave some variables with outliers untouched (mainly integer variables with very few outliers that were very close to the IQR), and we removed the outlier values for only four predictor variables, namely Front, Depth, Sale Price, and Total Living Area. For all of those, we combined the use of the IQR limits with some custom-set limits: both Front and Depth had some values close to 0 that the IQR didn’t catch, so we set a lower boundary of 5. In the case of Sale Price, the distribution looked similar to that of Total Value, for which we also decided to set the cutoff at $400.000. Lastly, we removed the outliers for Total Living Area whenever the value was under 400 (no one or two family dwelling should be expected to have only 37 m² living area), and kept the upper IQR limit as the higher bound for the cutoff.

Whenever we used the IQR values as cutoffs for outlier removal, we made sure to always use the IQR values of the training dataset to remove the outliers, even if we were removing the outliers of the training or test dataset.

3.3. Categorical Variable Encoding and Imputation

We chose to perform multivariate feature imputation to fill the missing values in our dataset using Sklearn’s IterativeImputer [7], which requires numerical data. It is an advanced imputation method that fills missing values based on information from all available features. This imputation method was especially important for the variable Sale Price which is highly correlated with the target and contained many missing values. We wanted the imputer to use information from the categorical variables, so we chose one-hot encoding (OHE) for our categorical variables over label encoding in order to avoid unwanted, artificial inferences in our imputation process caused by the numbers that are arbitrarily assigned to the different categories by label encoding. Because OHE avoids the assignment of arbitrary numbers to categories, it escapes any possible encoding-induced bias in the imputation [8].

We trained a Sklearn IterativeImputer using our training dataset without the “Total Value” variable to avoid incorporating information about the target or about the predictor variables, and to also ensure that no information from the validation and test datasets was incorporated in the training of the imputer. We then proceeded to impute the three partitions of our dataset using the imputer trained over the training dataset.

3.4. Feature Selection

After removing outliers and performing imputation, we studied the correlation between our numeric and the relation between categorical variables and the target variable in order to identify features that were important for explaining the variability of the target (figs. 1, S5, S7, and S8). It became obvious that the home's location is important because only a few neighborhoods contain properties with a total value over $300,000 while the majority of neighborhoods have median values less than $100,000. Two of the neighborhoods which have long tails, Kensington-Bailey and North Park, are also in the top three neighborhoods by property count, with approximately 14% of the city's one family and two family dwellings within those two neighborhoods.

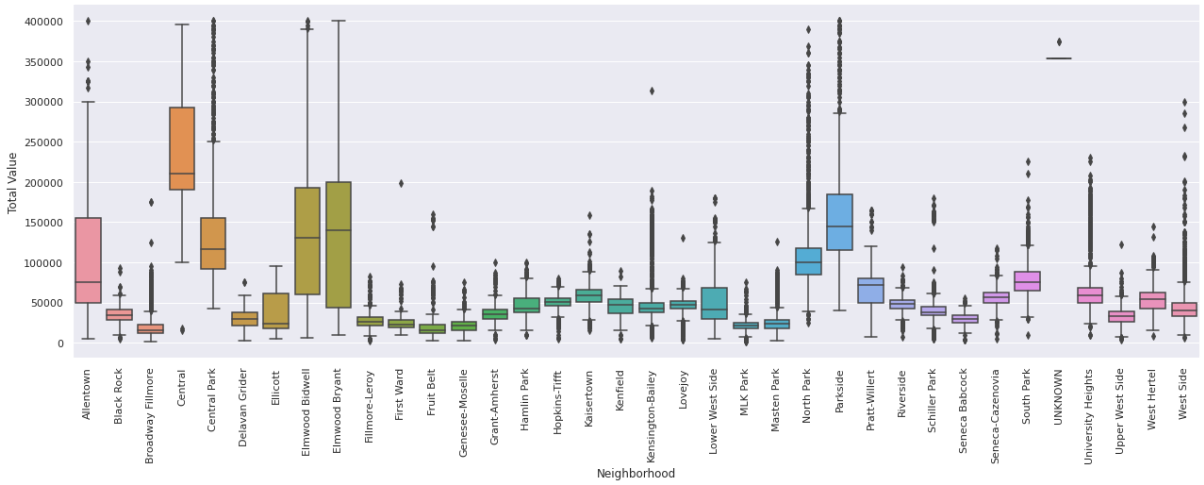

Figure 1. *Total Value boxplots by neighborhood.*

Another important categorical variable was property class description because each category had two different, approximately normal distributions for Total Living Area, which is very associated with the target variable, as can be seen in figure 2. Curiously, the distribution of the target of both One Family Dwelling and Two Family Dwellings are almost identical, as can be seen in figure S6. This means that taking into account the class of property we are looking at will be very important to estimate the target variable based on Total Living Area.

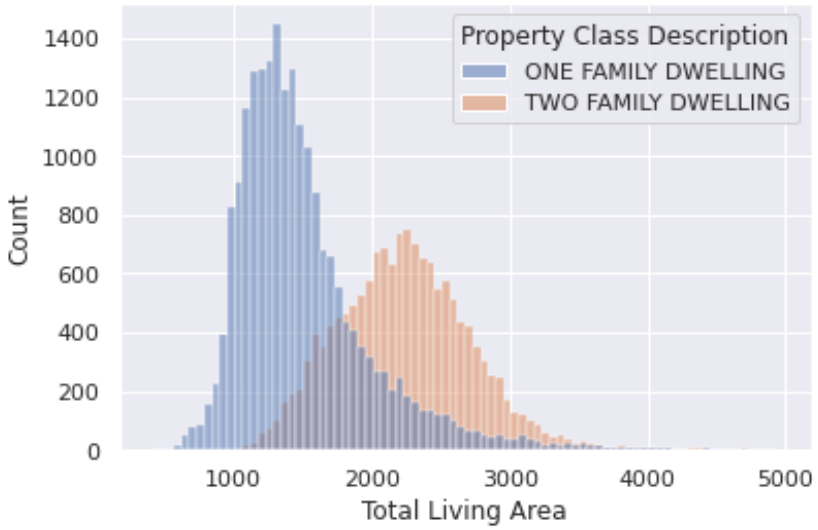

Figure 2. *Count histogram based Total Living Area, split in two series based on Property Class Description.*

We used this analysis to eliminate all those variables that were not linked at all to the target variable, as well as to select the set of most interesting features that we would later use to train our models.

3.5. Final Dataset

After running the whole preprocessing pipelines, the training, validation, and test dataset partitions contain 35921, 12020, and 16082 rows respectively, with a total of 21 variables.

4. Modeling

Since the target variable is numeric, we consider regression models. We have deployed two classes of models: linear regression models and decision tree (and tree ensembles) models, with instances of different types for each class.

4.1. Validation Strategy

Our validation strategy consisted of keeping three separate datasets: one for training the models, one for validating them, and a test dataset to study the generalizability and the performance of our final model. The proportions of those are approximately 50-20-30, respectively. The reason we chose this approach instead of k-fold cross validation is that our dataset contains a vast amount of samples, meaning that all of our dataset splits will be a good sample of the total population, and that using k-fold cross validation models would cause our training times to be too high.

4.2. Quality and Performance Metrics

Given that our models are regression models, we have chosen an adequate set of statistical metrics supported by Sklearn to help us understand the performance of the different models, shown in Table 1. We will focus mostly on Root Mean Squared Error (RMSE) and the Coefficient of Determination (R2 score), which we want to minimize and maximize, respectively. The RMSE allows us to evaluate how different the model's predictions are from the target values using the same units as the target variable which makes it easier to interpret than the Mean Squared Error. The R2 score represents the amount of variance in the target that can be explained by the features included in the model, and can be an indicator of the model's goodness of fit and how well it will generalize to unseen data [9].

Table 1. *Model performance metrics chosen.*

Mean Squared Error (MSE)
Root Mean Squared Error (RMSE)
Mean Absolute Error
Median Absolute Error
R2 Score

4.3. Linear Regression Modeling

Linear regression was chosen as the base model because of its simplicity, interpretability, and its quick training process. We have tried four different linear regression models, namely Ordinary Least Squares regression (OLS), Lasso regression, Ridge regression, and Huber regression. Here's a list with the reasons for which we have chosen each of these types of model:

- **OLS regression**: Basic, simple model with which to touch ground and explore the dataset.
- **Ridge regression**: Introduces regularization by penalizing the magnitude of the coefficients.
- **Lasso regression**: Introduces regularization and allows for automatic feature selection.
- **Huber regression**: Provides robustness in the presence of outliers.

For each type of model, we used Sklearn's default parameters. We trained each model with three different sets of features, making it a total of 12 linear regression models. We tracked the performance metrics and training time for each model. The categorical features were pre-processed using OHE, and no transformations were done on the numeric variables. The three sets of features can be consulted in the following list:

- **Numeric set:** 6 numerical variables: Front, Age, Total Living Area, # of Fireplaces, # of Beds, TotalBaths. 1 categorical variable: Property Class Description.
- **Limited set**: Numeric set + 1 categorical variable: Neighborhood.
- **Extended set**: Limited set + 3 categorical variables: Overall Condition, Building Style, Heat Type.

4.3.1. Results

We have found that overall, the limited set of features performed better than the numeric set, but adding more categorical features beyond the limited set did not improve the model significantly. This indicates that the neighborhood is an important categorical variable, because the target variance explained by the model improved by approximately 20% when it was added. The results for the different linear models can be consulted in table 2.

OLS and Lasso regression models had very similar performances, while the Ridge regression model had the worst performance. When comparing the results between the training and validation partitions, we see that each model performed consistently when it was evaluated with the validation partition, which indicates that none of the models have overfit to the training data, and also that the training set is representative of the general population.

In order to choose our preferred model, we take into account all of the performance metrics, the complexity of the model, and the training time. The OLS model trained with the Limited set of features is the preferred model of Linear Regression.

Table 2. *Linear regression model performance results. The best validation results for each model are shown in bold.*

Model	Features	Partition	MSE	RMSE	Mean Abs. Error	Median Abs. Error	R2 Score	Train Time (seconds)
OLS	Numeric Set	Train	1.23E+09	35,068	24,645	17,785	0.55	0.05
OLS	Numeric Set	Validation	1.18E+09	34,380	24,240	17,169	0.56	0.05
OLS	Limited Set	Train	6.64E+08	25,773	16,066	10,006	0.76	0.21
OLS	**Limited Set**	**Validation**	**6.34E+08**	**25,187**	**15,809**	**9,991**	**0.76**	**0.21**
OLS	Extended Set	Train	6.49E+08	25,480	15,852	9,880	0.76	0.46
OLS	Extended Set	Validation	6.22E+08	24,940	15,622	9,862	0.77	0.46
Lasso	Numeric Set	Train	1.23E+09	35,068	24,645	17,782	0.55	0.04
Lasso	Numeric Set	Validation	1.18E+09	34,380	24,239	17,171	0.56	0.04
Lasso	Limited Set	Train	6.64E+08	25,773	16,066	10,008	0.76	5.78
Lasso	**Limited Set**	**Validation**	**6.34E+08**	**25,188**	**15,809**	**9,990**	**0.76**	**5.78**
Lasso	Extended Set	Train	6.49E+08	25,479	15,857	9,876	0.76	13.41
Lasso	Extended Set	Validation	6.22E+08	24,936	15,625	9,845	0.77	13.41
Ridge	Numeric Set	Train	1.23E+09	35,068	24,645	17,783	0.55	0.02
Ridge	**Numeric Set**	**Validation**	**1.18E+09**	**34,380**	**24,240**	**17,171**	**0.56**	**0.02**
Ridge	Limited Set	Train	1.39E+09	37,277	26,040	18,897	0.5	0.07
Ridge	Limited Set	Validation	1.35E+09	36,751	25,782	18,851	0.5	0.07
Ridge	Extended Set	Train	1.38E+09	37,132	25,954	18,806	0.5	0.09
Ridge	Extended Set	Validation	1.34E+09	36,623	25,719	18,819	0.5	0.09
Huber	Numeric Set	Train	1.30E+09	35,987	24,340	16,920	0.53	0.78
Huber	Numeric Set	Validation	1.25E+09	35,337	23,949	16,492	0.53	0.78
Huber	Limited Set	Train	1.14E+09	33,774	20,650	12,805	0.59	0.94
Huber	Limited Set	Validation	1.09E+09	33,066	20,216	12,471	0.59	0.94
Huber	Extended Set	Train	9.75E+08	31,233	19,186	11,696	0.65	1.24
Huber	**Extended Set**	**Validation**	**9.38E+08**	**30,625**	**18,843**	**11,583**	**0.65**	**1.24**

From the scatter plot comparing the predictions to the target (fig. 3 LHS), we see that our best linear model, the Lasso regression with the limited set of features, has difficulty predicting total values over $300,000 even though the training data contains samples with total values up to $400,000. This is consistent with the data visualization by neighborhood, which showed all except one neighborhood with median property values less than $150,000, but certain neighborhoods have long tails of outliers.

The plot of residuals is close to a normal distribution with the mean close to 0 which indicates the linear model works well (fig. 3 RHS). The tails are long which indicates that there are outliers present that the model does not predict well.

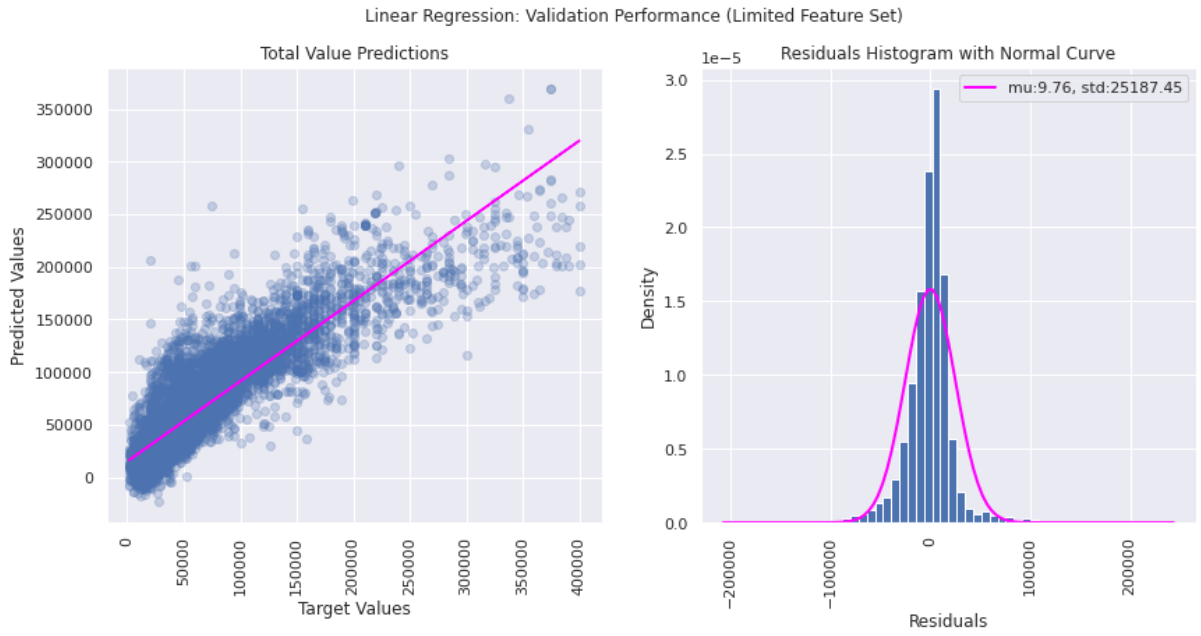

Figure 3. *Visualizations for the best linea model: The Lasso Regression with the limited set of features. (LHS) Scatter plot comparing the real target values with the predicted ones. (RHS) Histogram with residual values compared to the normal curve stemming from their mean and standard deviation. The mean is higher than 0, meaning that the model has a tendency to overestimate Total Value.*

4.4. Decision Tree and Random Forest Modeling

Decision trees and random forests were chosen for a second set of models. Decision trees and random forests benefit from a good interpretability, have short and medium training times, respectively, and are robust regression methods that have previously proven to be good candidates for modeling real estate sale prices [10].

In a similar manner as we had done before, we have trained our decision tree models with two sets of variables. However, this time, the extended set of variables contains a higher number of features. We made this decision because while linear models struggle when dealing with collinearity, decision trees and random forests have the means to deal with the problems stemming from intercorrelated variables. This means that we don't need to select our features so carefully, at the expense of longer training times. After seeing the superior results of the extended set in our first model, we decided to only deploy random forest models using the extended set. The two sets of variables used for the tree and forest models can be consulted below:

- **Limited set**: Contains only numerical variables, namely: Property Class Description, Neighborhood, Front, Age, Total Living Area, # of Fireplaces, # of Beds, TotalBaths.

- **Extended set**: Limited set + Depth, Area, Sale Price, Latitude, Longitude, Property Class Description, Overall Condition, Building Style, Heat Type, Basement Type, Neighborhood, SameOwnerLastName, OwnershipType, PropertyClassReformed, Sale Type, Age.

To further improve the performance of our tree and forest models, we decided to exploit sklearn's GridSearchCV function to explore different parameter sets for both. In table 3, we

present the different parameters that we introduced in GridSearchCV for each of the model classes.

Table 3. *Model parameters introduced in GridSearchCV for the tree and forest models*

Decision Tree parameters *(192 fits)*		**Random Forest parameters** *(32 fits)*	
cv	3	cv	2
splitter	[“best”,”random”]	bootstrap	[True]
max_depth	[None, 30]	n_estimators	[50, 200]
min_samples_leaf	[1, 4]	min_samples_leaf	[3, 6]
min_samples_split	[1, 4]	min_samples_split	[3, 6]
max_features	["auto","log2","sqrt",None]	max_features	["auto","sqrt"]

4.4.1. Results

The results of all our decision tree and random forest models can be found in table 4. The first decision tree, built using only the limited set of features has a performance only slightly below that of the best linear models, which without further optimization seemed a good starting point. This first tree is extremely overfitting, with a very low RMSE and a R2 score of 1, but still has a decent performance over the validation dataset.

Table 4. *Tree and forest models’ performance results. Best validation results for each class shown in bold.*

Model	Features	Test Partition	MSE	RMSE	Mean Abs. Error	Median Abs. Error	R2 Score	Train Time
Tree 1	Limited Set	Train	1455887.0	1206.6	159.47	0.0	1.0	00:00:06.7
Tree 1	Limited Set	Validation	789999210.63	28106.92	14497.64	7000.0	0.71	00:00:06.7
Tree 2	Extended Set	Train	0.0	0.0	0.0	0.0	1.0	00:00:15.2
Tree 2	Extended Set	Validation	900271796.19	30004.53	14465.08	6400.0	0.66	00:00:15.2
Tuned Tree	Extended Set	Train	102724021.89	10135.29	5013.02	2342.86	0.96	00:02:33.8
Tuned Tree	**Extended Set**	**Validation**	**757704265.32**	**27526.43**	**13566.1**	**6326.79**	**0.72**	**00:02:33.8**
Forest	Extended Set	Train	35890606.89	5990.88	3028.33	1416.0	0.99	00:09:19.7
Forest	**Extended Set**	**Validation**	**588804999.42**	**24265.3**	**11504.57**	**5088.5**	**0.78**	**00:09:19.7**
Tuned Forest	Extended Set	Train	82196423.64	9066.22	4381.9	1971.15	0.97	00:40:44.9
Tuned Forest	Extended Set	Validation	627226800.21	25044.5	11797.09	5068.16	0.77	00:40:44.9

The second tree, trained with the extended set of features, demonstrated a more accentuated overfitting, with a RMSE of 0. The performance over the validation dataset was significantly worse, with a lower R2 score and a much higher RMSE than the first tree.

Once we arrived at this point, it was clear that enforcing a new training method with new parameters to avoid overfitting was necessary. To this end, we implemented the use of GridSearchCV and found a set of parameters that yielded a third tree, an improved model that is better at generalizing. The different parameter sets explored with GridSearchCV to train the tuned Tree model can be consulted in table 5.

The tuned tree model yielded much better results: the model doesn't overfit the training data, as the more realistic RMSE and R2 score show, and it does a better job at generalizing, as the results over the validation data demonstrate: the RSME is the lowest of all the tree models, and the R2 score is the highest.

After deploying the three tree models, we decided to train and test some random forest models. Because ensembles of a given type of model have the tendency to perform better than the best instances of the models, we hoped to develop a more accurate model that is better at generalization.

We first generated a random forest model with the default values, and we indeed obtained better results than with our best tree. The RMSE was reduced significantly, while the R2 of the model increased to 0.78.

However, we still found that the forest model was overfitting the training data, for which we also attempted to use GridSearchCV to come up with a set of parameters other than the default one, with the hope to find a model that is better at generalizing. We explored 16 sets of parameters, with a 2-fold cross-validation strategy to reduce the number of fits and therefore the training time. The parameters for the best model can be found in table 5.

Table 5. *Optimal model parameters found by GridSearchCV for the tree and forest models*

Decision Tree - Optimized parameters		**Random Forest -** Optimized parameters	
splitter	"best"	bootstrap	True
max_depth	None	n_estimators	200
min_samples_leaf	4	min_samples_leaf	3
min_samples_split	4	min_samples_split	6
max_features	None	max_features	"auto"

The resulting tuned forest model seemed to overfit the training data less than the first forest model, but it performed marginally worse than the linear model when tested with the validation dataset.

4.5 Model Decision

After considering all the deployed linear regression models, and decision tree and random forest models, together with their results over the train and validation datasets, we have chosen the first random forest model as our final model. Its marginally better performance with a smaller number of estimators (100 in the forest model vs 200 in the tuned forest

model) makes it a better choice, given that it's a simpler model with nearly the same performance.

5. Final Model Testing and Results

After selecting our final model, we tested it using the test data partition, which hadn't been used either to train or to validate the models. This testing will allow us to properly assess the generalization capability of our model, by studying the model's performance over unseen data.

The results obtained after using the model to predict the prices for the test dataset can be consulted in table 6, together with the results obtained previously when using the train and validation sets. We were surprised to find that the model performed much better over the test partition than it had done over the validation partition. The RMSE was much lower than we had observed before and the R2 score reached 0.86, resulting in a seemingly great model.

Table 6. *Final model results.*

Model	Features	Test Partition	MSE	RMSE	Mean Abs. Error	Median Abs. Error	R2 Score	Train Time
Forest	Extended Set	Train	3.58e+07	5990.88	3028.33	1416.0	0.99	00:09:25.3
Forest	Extended Set	Validation	6.06e+08	24624.19	11604.10	5103.00	0.78	00:09:25.3
Forest	Extended Set	Test	3.69e+08	19215.82	10134.03	4735.00	0.86	00:09:25.3

This finding made us question our protocol. Had we not shuffled the data before performing the split? Could some sort of bias have been introduced in the split because of some pattern in the order of our data? We backtraced our steps and went over the documentation of the functions used to split the data. As we thought before running the model over the test data, we did enforce shuffling, for what the validation and test samples should be representative samples of the total population. However, the sizes of the three partitions (all well over 10.000 rows) make it quite improbable that the results over the validation and the test sets differ so much. We haven't found a good explanation for this difference, beyond the fact that the test partition population seems to be more similar to the training partition than the validation partition population was.

When exploring the results more in depth, we have found some other curious effects. As can be seen in figure 4, our model systematically underestimates the property prices. The population of predicted target values deviates from the real target values for values over $150.000. Over this value, the model's tendency is to underestimate the target's predictions, never reaching the highest values in the real target population (of almost $400.000), being the highest predicted price of approximately $350.000.

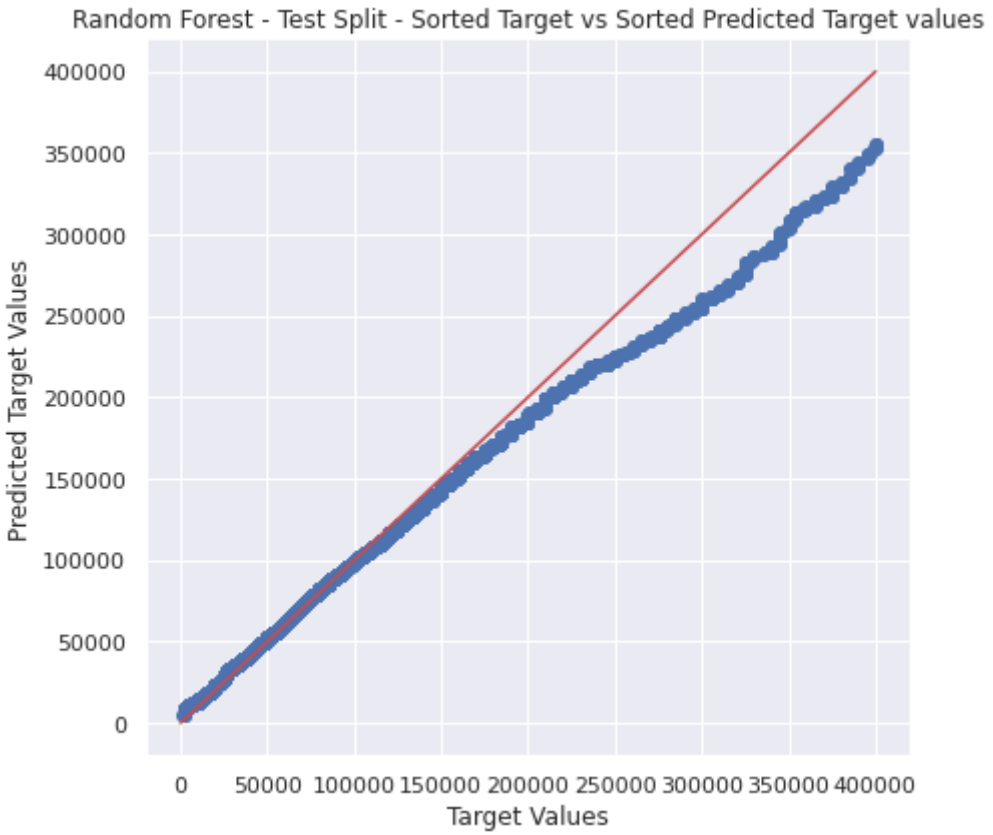

Figure 4. *Comparison of the predicted and real target value populations for the final model results over the test dataset. The real, sorted values of the target are plotted against the X axis, while the predicted, sorted values of the target are plotted against the Y axis. A red diagonal line marks the expected path that the points should follow if both populations were equivalent.*

After noticing this effect, we suspected that our cutoff choice for removing target variable outliers could have caused it. To test whether selecting an outlier threshold above the 200.000k limit obtained by applying the 3-fold interquartile range rule had an impact on our model, we decided to run it again with a version of the dataset where the threshold for removing rows based on target variable outliers is set at the “recommended” 200.000k and not 400.000k. However, after running the model, we observed a similar effect: the higher half of the predicted values were, just like in figure 1, lower than the real values in the higher half. It seems that the way in which we have trained our model leads to underestimating bias in the predictions of the target variable.

Overall, it seems like our final model has a rather good performance over new data. Even if the validation results were slightly worse than the results over the test partition, the test partition contained approximately 16,000 properties. Our model was capable of explaining 86% of the variance in the target variable and has an median error of approximately $4600, which is less than a 10% of the median “Total Value” value ($49000). The model can be considered to be quite robust, even if there’s still margin to improve.

6. Conclusions

We have been able to develop a non-linear model that is able to generalize and predict our target variable significantly well. However, as with any machine learning model developed,

it's key to understand its context and its limitations. Because the tax that a property owner must pay is based on the Total Value of the property, calculating this value is a quite delicate task. We can also assume that the value of a property may be strongly associated with some variables that are intangible to us given the original dataset we have worked with. A building's orientation, the amount of sunlight that it receives, or the kinds of businesses and services that surround it, are all variables that can heavily influence the value of a property [11].

We believe that, while our model seems to perform quite well given the scope of the project, it's far from being a model that could be deployed into production. Without adding further information to the dataset, there's much margin to improve. We haven't managed to find a set of parameters for our best models (trees and forests) that don't cause the model to overfit the training data. It's possible that finding a parameter set that enables the training of a model that performs as well over training data as it does over validation data, we could find a model that is able to generalize better than ours. We have also shown that our selected model struggles to produce a prediction that fully resembles the population of the real target variable, systematically underestimating the value of the more expensive properties. This would imply that the owners of the more expensive buildings would be taxed less, potentially having an impact over the cities' public funds.

Therefore, further refinements of the models we have deployed would be critical, as well as the exploration of other types of models, such as support vector regressors, which have previously been shown to be good at predicting real estate prices, as commented before [6].

In addition to more advanced modeling, it may be necessary to collect data about features such as proximity to schools, parks, and medical facilities, hours of sunlight, or property orientation. If collecting additional data to improve the model is not possible, separate models may be needed for different locations rather than relying on a single model for the whole city based on property class.

Beyond this, our model is restricted to properties under $400.000, which are the vast majority of properties found in Buffalo, NY. However, more expensive properties will also need to be assigned a value. When attempting to model the more valuable properties, it would be key to consider that modeling luxury properties can be a much more complex problem. Those properties tend to be subject to economic processes more linked to stock markets than to common real estate operations [12], meaning that the reality explaining their value could completely escape that reflected by the variables collected by Buffalo's financial authorities, and those proposed above by us.

In conclusion, a deeper understanding of market knowledge in the real estate industry could improve the modeling process, especially in the parts related to feature engineering and feature selection. A larger, multi-disciplinary team would likely be beneficial to the outcome of a project developing machine learning models for property value assessment during tax rolls.

7. Future Applications

Due to the complexity of the original dataset, we made several choices during pre-processing in order to simplify the dataset to develop models that could generalize well

to the majority of properties on the Buffalo tax assessment roll. These models are limited in their application to the properties in the categories of one family dwelling or two family dwelling that are not considered outliers for the target variable.

There are some other factors that could improve the model that we haven't explored. Variable transformations, both over the predictor variables and the target variables, could have a huge impact on the performance of the models. Enriching our dataset with polynomic transformations of the different numeric variables could be a good next step if we were to continue developing our models.

Future work could include extending the models to include more property classes, developing separate models to handle the more expensive properties, therefore exploring the full complexity of the source data. And applying the models to cities other than Buffalo, where the features driving the value of a property could differ significantly, would also be interesting.

As we have commented before, our models have lots of room for improvement, and continuing our experimentation would doubtlessly yield a very performant model. However, because we have focused mostly on developing a good data preparation and model validation pipeline, trying to introduce all the good practices we have been taught about during our theory and practice sessions, and not so much in developing the most optimized model possible, we can say that we are quite satisfied with our work.

References

Cover picture reproduced from Visit Buffalo Niagara (accessed Jun. 10).

1. City of Buffalo, “Assessment & Taxation Department | Buffalo, NY.,” buffalony.gov, https://www.buffalony.gov/187/Assessment-Taxation-Department (accessed Jun. 10, 2022).
2. City of Buffalo, “2019-2020 Assessment Roll | Internal | Open Data Buffalo.,” buffalony.gov, https://internal.data.buffalony.gov/Government/2019-2020-Assessment-Roll/kckn-jafw (accessed Jun. 10, 2022).
3. Jácome González, M. and K. Weissman, “KatBCN/ML-PropertyAssessment.,” https://github.com/KatBCN/ML-PropertyAssessment (accessed Jun. 10, 2022).
4. Zillow Group, “Zillow Prize Winners.”, https://www.zillow.com/z/info/zillow-prize/ (accessed Jun. 10, 2022).
5. García-Magariño, C. Medrano, and J. Delgado, “Estimation of missing prices in real-estate market agent-based simulations with machine learning and dimensionality reduction methods,” Neural computing & applications, vol. 32, no. 7, pp. 2665–2682, 2020, doi: 10.1007/s00521-018-3938-7.
6. Scikit-learn developers, “sklearn.model_selection.train_test_split.,” scikit-learn.org, https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html (accessed Jun. 10, 2022).
7. Scikit-learn developers, "6.4. Imputation of missing values.," scikit-learn.org, https://scikit-learn.org/stable/modules/impute# (accessed Jun. 10, 2022).
8. Yadev, Dinesh, “Categorical encoding using Label-Encoding and One-Hot-Encoder.,” https://towardsdatascience.com/categorical-encoding-using-label-encoding-and-one-hot-encoder-911ef77fb5bd (accessed Jun. 10, 2022).
9. Scikit-learn developers, “3.3.4. Regression metrics.,” scikit-learn.org, https://scikit-learn.org/stable/modules/model_evaluation.html#regression-metrics (accessed Jun. 10, 2022).
10. Fan, Gang-Zhi, et al. “Determinants of House Price: A Decision Tree Approach.” Urban Studies, vol. 43, no. 12, Nov. 2006, pp. 2301–2315, doi:10.1080/00420980600990928.
11. Fleming, David and Grimes, Arthur and Lebreton, Laurent and Maré, David C. and Nunns, Peter, Valuing Sunshine (June 29, 2017). Available at SSRN: https://ssrn.com/abstract=2997312 or doi: http://dx.doi.org/10.2139/ssrn.2997312
12. Bloomberg L.P. “The Big, Big, Big, Big Money Behind Tall Buildings.” Link: https://www.bloomberg.com/news/articles/2015-06-11/how-luxury-condos-became-a-major-global-investment-tool Bloomberg Terminal. June 12, 2015.

Appendix

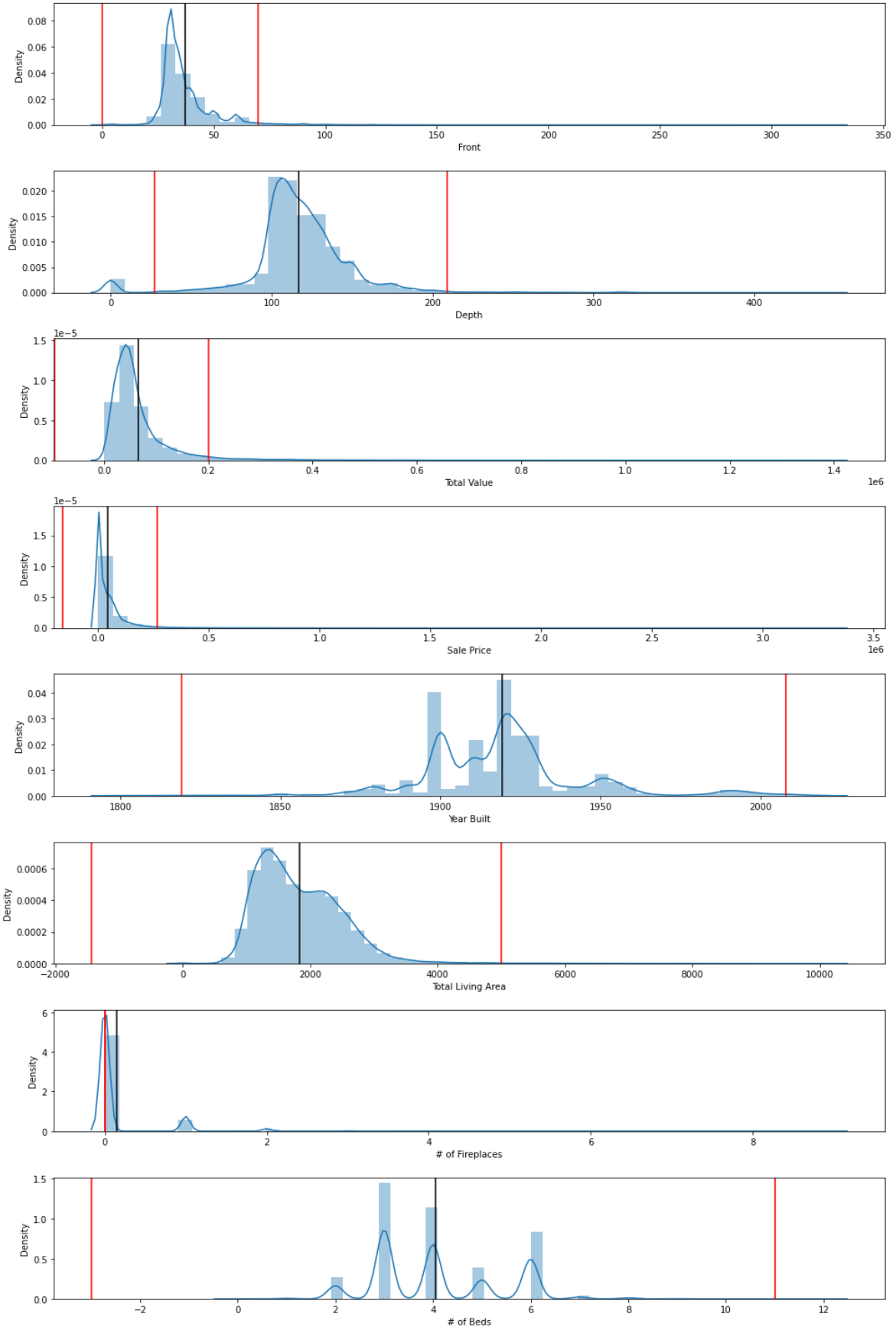

Figure S1. *Histograms for the different numerical variables. The red vertical lines indicate the lower and upper IQR limits.*

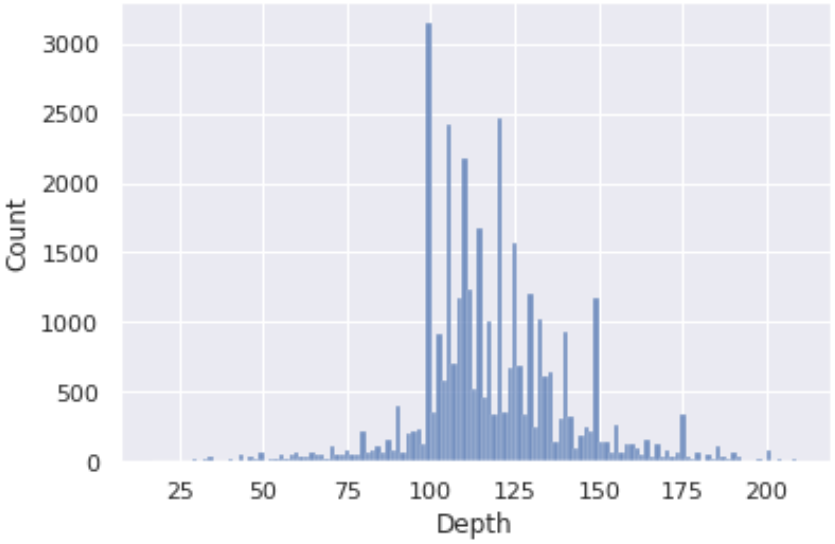

Figure S2. *Count histogram based on Depth*

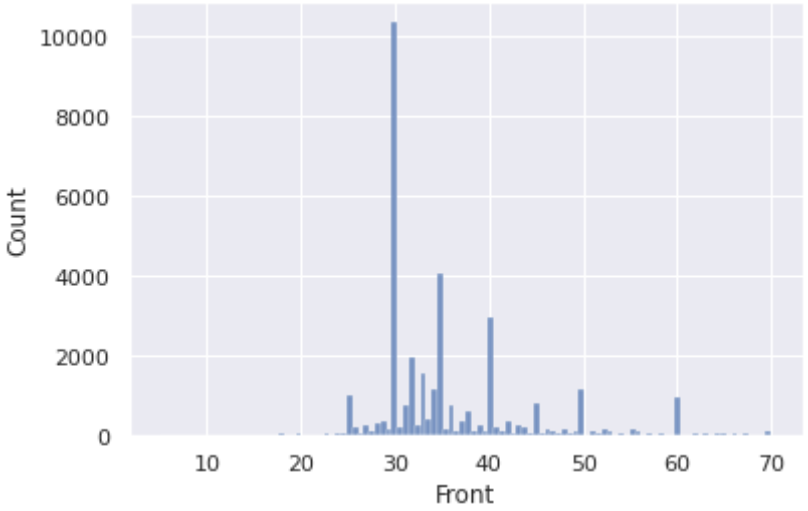

Figure S3. *Count histogram based on Front*

Figure S4. *Count histogram based on Sale Price*

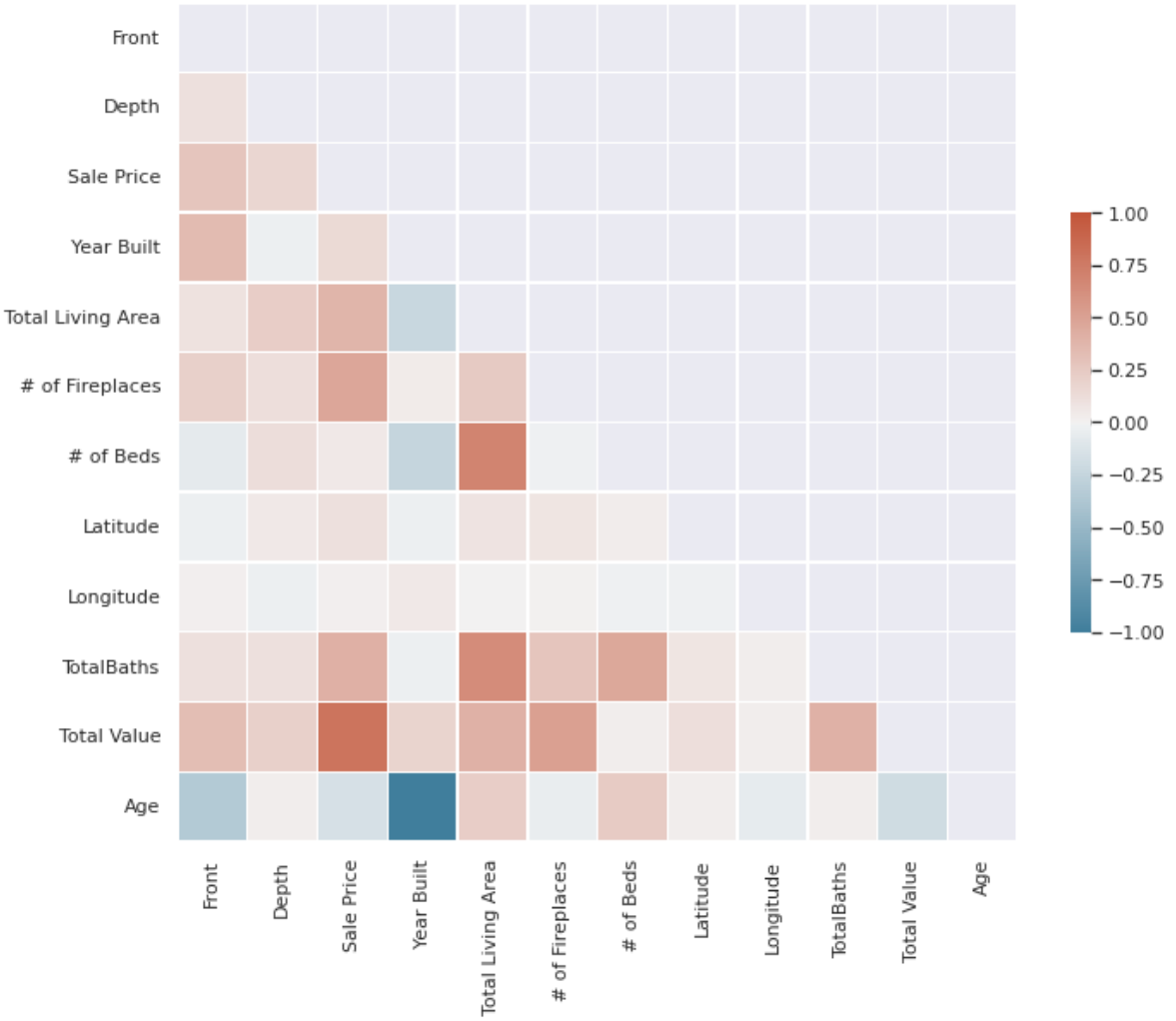

Figure S5. *Correlation Plot for Numeric Variables.*

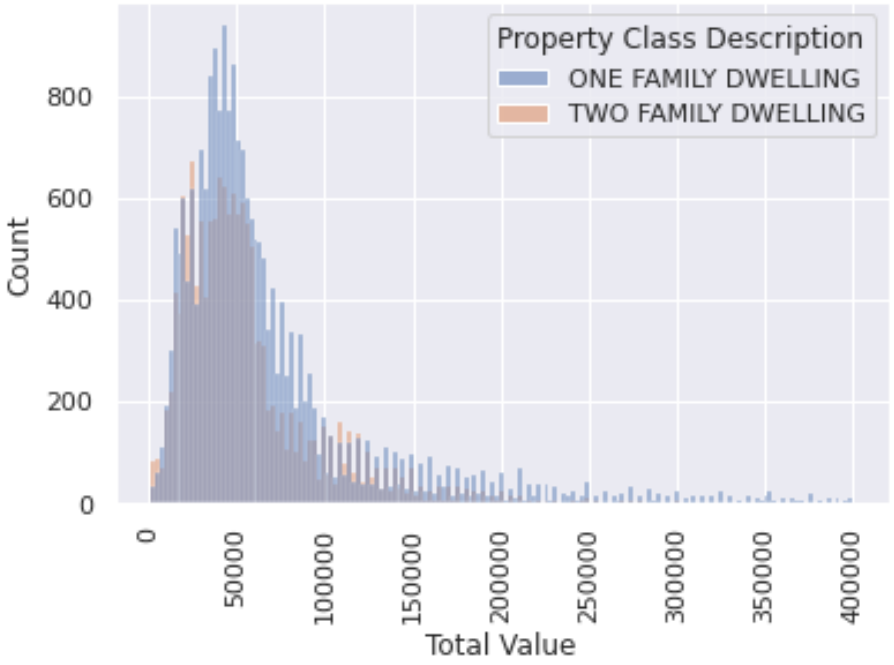

Figure S6. *Count histogram based Total Value, split in two series based on Property Class Description.*

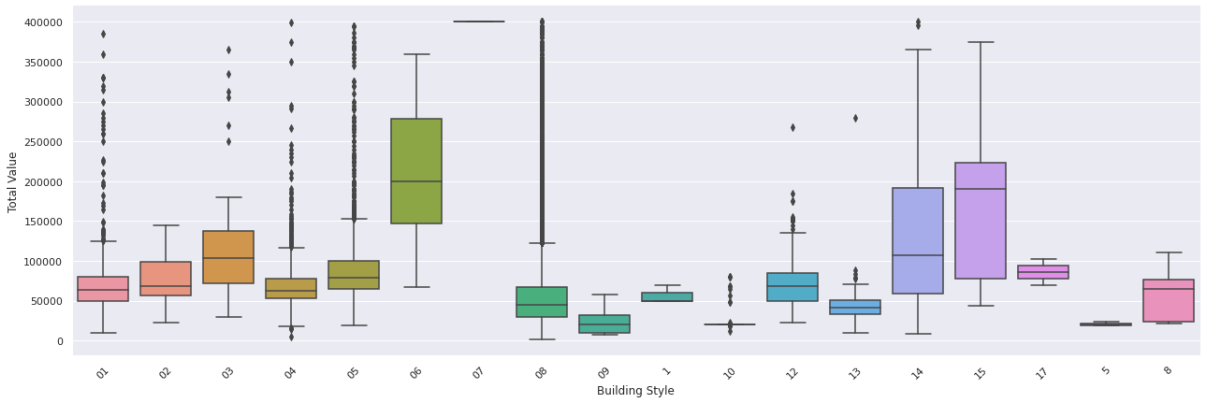

Figure S7. *Target variable box-plot by building style.*

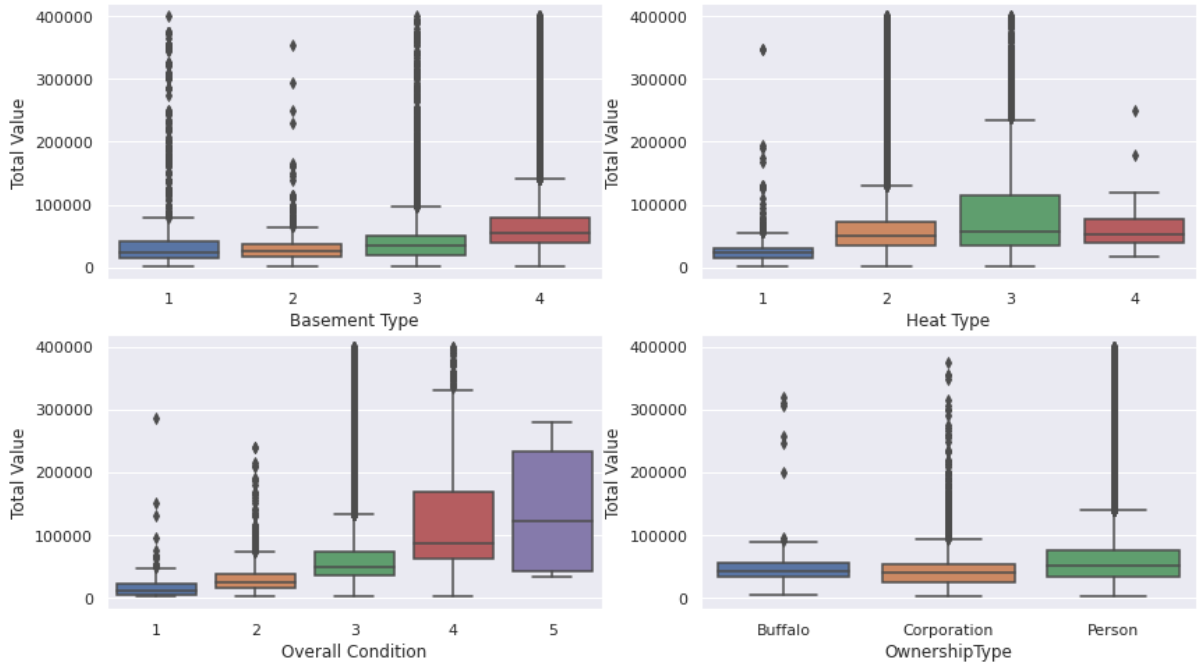

Figure S8. *Target variable box-plot by Basement Type (top left), Heat Type (top right), Overall Condition (bottom left), and OwnershipType (bottom right).*