Kevin Maher

Prof. Geena Kim

MSDS692 – Data Science Practicum

June 25, 2017

Predicting the Sales Prices of Boulder County Homes

Introduction:

The objective is to predict the sales prices for single family homes in Boulder County, Colorado by using data from the county assessor's office. The data is for home sales from 2013 to 2016. The problem is inherently a regression problem, so linear regression plus a number of other machine learning regression methods will be experimented with in an effort to produce as accurate a model of housing prices as possible.

Only detached single family homes are considered in this model. This is because of constraints imposed by the computing power available and the model size. In the four year period studied over 14,000 detached single family homes were sold in the study area. This makes for a large model and it was not feasible to add other housing types such as condominiums and mobile homes and still have a tractable model for the semester project.

Data Description:

The data from the assessor's office consists of Excel files for each year's sales and additional files for different property characteristics. These data files are provided to student's at no cost by the assessor's office, but normally they charge a fee for the data. Accordingly, the raw data has not been made available with the project report and scripts on the GIT website (GitHub).

Data for each year's sales need to be matched to the various property characteristics tables for each matching year. In essence each file may be regarded as a SQL table where the key is the property account number. Joins are made using the account number between tables for the same year so that a property sale may be matched with the condition or features of the property using data for the appropriate year.

All of the Excel files were converted to 'csv' files because the “xlsx” files were not compatible with my version of OpenOffice. This was done in bulk using a Python script (xl_to_csv.py). I chose to create a single sales table for the years 2013 to 2016. This was done by stacking the “csv” files using OpenOffice Calc. This file was the cleaned for data analysis using a Python script (sales_clean.py). Only data with a sales code of 'Q' was kept as the assessor's office indicates that these are given as arm's length transactions where money is exchanged (Boulder County Assessor Public Dataset). Also, only property coded as “Real Property” was kept as homes are a type of real property. Because home prices are expected to be time sensitive, the year and month of the sale were extracted in this script, a time period feature was later extracted from the year and month.

The sales table thus provided the following data. The account number called “strap” in the file headers, this is the primary key by which tables are joined. The sales table also provided the deed type, sales price, year and month of the sale. This was all of the data that appeared to be useful from the sales table. The sales table was used as the main starting table, data from the other tables was then merged in to create a full data set for each property.

The main data setup and table joins were done in a Python script (Assemble_Data.py). This script extracted the data for each year and merged in the data from the property data tables. These tables included data for the assessor's valuations, land data, building data and property

location. Except for the building data these tables have a one to one relationship so the merge was straightforward. For the buildings table there is the possibility of multiple buildings for a single property. In order to simplify the model and make it tractable for the time available, only the primary building for each property was considered in the model. The numbers and types of secondary buildings are complex and there was not time to properly analyze this data in the semester framework. It is especially difficult because the building data is spread over four years and is for all properties in the county, it would need to be merged with the sales data and the categories of secondary buildings analyzed. If the project were continued in a second practicum then one possible method for enhancing the model would be to add the code needed to analyze the tables for secondary buildings. This might help especially with the more expensive homes where the model appears to become less accurate.

Another table with a many to one relationship to the sales table was the “other areas” table. This was analyzed and categorized for each property since a home may have multiple “other areas”. These are decks, patios, porches and other areas considered as valuable by the assessor. Primary categories in this table were decks, enclosed porches, porches and patios. For the rarer classes other than these it is difficult to tell how many examples will wind up in the sales data because the tables are for all properties in the county, including those that have not sold. To simplify the model and ensure an adequate number of values in the category, other areas which appeared to be too few in number to be considered individually were lumped together as “other areas”.

The table for other areas includes data for every property in the county, the data needs to be extracted for properties that sold during the year. The data also needs to be aggregated, for example a property might have multiple decks or patios. The Python script (Assemble_data.py)

takes care of this aggregation. For example, a home with two decks is shown as having decks (a Boolean value) and the area of both decks is added together in the square footage column for deck area. Other areas in the table are handled in a similar manner, multiple patio areas are aggregated into one area for example.

Numeric variables are shown in figure 1. These are mostly for various areas of the home or counts of the number of rooms of a given type. Area features include lot size, total finished interior area of the home, car storage and basement size. Counting features include the number of rooms that are not bathrooms, counts of bathrooms by type and the number of bedrooms.

Numeric Variables		
Variable Name	Description	Units
totalActualVal	Assessor's estimate of value used for setting taxable value	dollars
GIS_sqft	lot size (land)	square feet
bsmtSF	basement size	square feet
carStorageSF	car storage size	square feet
nbrBedRoom	number of bedrooms	count
nbrRoomsNobath	number of rooms that are not bathrooms	count
mainfloorSF	main floor size	square feet
nbrThreeQtrBaths	number of ¾ baths	count
nbrFullBaths	number of full baths	count
nbrHalfBaths	number of half baths	count
TotalFinishedSF	total finished size of the home	square feet
Deck_Area	total of deck atrea	square feet
Encl_Porch_Area	total of enclosed porch areas	square feet
Patio_Area	total of patio areas	square feet
Porch_Area	total of unenclosed porch areas	square feet
Other_Area	other areas, total, too few in overall number to break out	square feet

Figure 1. Summary of Numeric Variables

Categorical and Boolean variables also were considered in the model (see figure 2). These give the presence of home features such as decks, patios and porches, construction and location information. Heating and air conditioning are categorized, as are car storage type and

codes for basement type. Car storage may be an attached or detached garage, or a carport for example. Basements may be entirely below ground, or be partially exposed above ground on sloping lots, thus producing a "walk out" basement. For the multi-category features, dummy Boolean variables were created.

Categorical Variables	
Has_Deck	home has one or more decks
Has_Encl_Porch	home has one or more enclosed porches
Has_Patio	home has one or more patios
Has_Porch	home has one or more open porches
Has_Other_Area	home has one or more other areas considered valuable
designCodeDscr	design code, such as single or multi-story
qualityCode	a quality rating for the property
bsmtType	type of basement, walkout for example
carStorageType	type of car storage
AcDscr	type of air conditioning, or none
HeatingDscr	type of heating
Has_Asbestos	home has asbestos siding
ExtWallDscrPrim	primary exterior wall description
ExtWallDscrSec	secondary exterior wall description
IntWallDscr	interior wall description
Roof_CoverDscr	roof covering description
city	city the home is in, or unincorporated, or area such as mountains

Figure 2. Categorical and Boolean Variables.

Exploratory Data Analysis:

Exploratory data analysis shows that by far the most important variable is the assessor's own estimate of the property's value. It may seem questionable to include this in the model, but the assessor's estimate is available from the data prior to the sale of each home. Any Realtor trying to price a home for sale could gain access to this information, as could any mortgage broker or insurance agent trying to value a home or determine if a given sales price was reasonable.

The predictive power of the assessor's estimate of the property's value can be shown with a linear regression. This produces an R-squared adjusted of 0.8937 using the script "value_vs_price.py". While the R-squared adjusted is high, other statistical measures show that there are problems using this variable as the sole predictor. The mean absolute percent error is 16.4%. Also, the gradient of the fit line is 1.13, indicating that the fit line tends to predict a lower sales price than the actual sales price.

Figure 3. shows the result of a linear regression of these two variables on what will be the test set for the machine learning model. The orange regression line was generated using the data, the green line shows what would be the result is the fit were perfect. The assessor's estimate under values homes, they tend to sell for more than estimated, as witnessed by the orange regression line being above the green line of perfect fit. Both the intercept and the gradient of the orange fit line show this problem where the assessor's estimate under estimates home values.

The goal of the machine learning exercise will be to improve on the predictive power of this single variable and linear regression by using more predictors and more advanced machine learning techniques. Since the mean absolute percent error is quite high, the gradient is too high along with the intercept, it is the goal that machine learning will improve upon these metrics.

Figure 3. Assessors Estimate vs Actual Sales Price

Another possibly important predictor of a home's sales price is the size of the house, this is primarily expressed in the Boulder County Assessor's data as the feature “TotalFinishedSF”. A plot of this relationship is shown in figure 4. It appears to show that this feature will add some predictive power to the model. A second metric for a home's size also exists in the data as the home's main floor size in square feet. This also shows some correlation to price (see figure 5). Additionally the relationship between a home's basement size and price can be plotted. This shows that a significant number of homes do not have a basement. The plot also shows a slight tendency for prices to increase as basement size increases (see figure 6).

Figure 4. Total Finished Sq. Ft. vs Sales Price

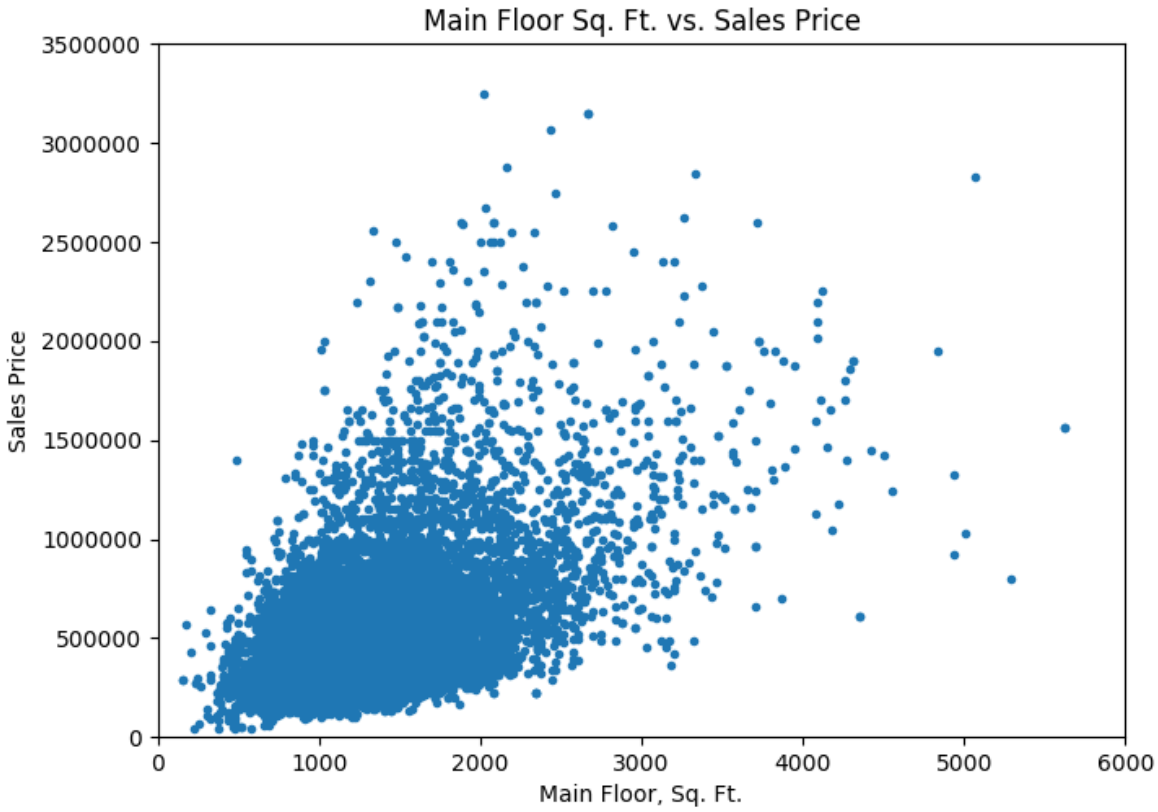

Figure 5. Main Floor Sq. Ft. vs Sales Price

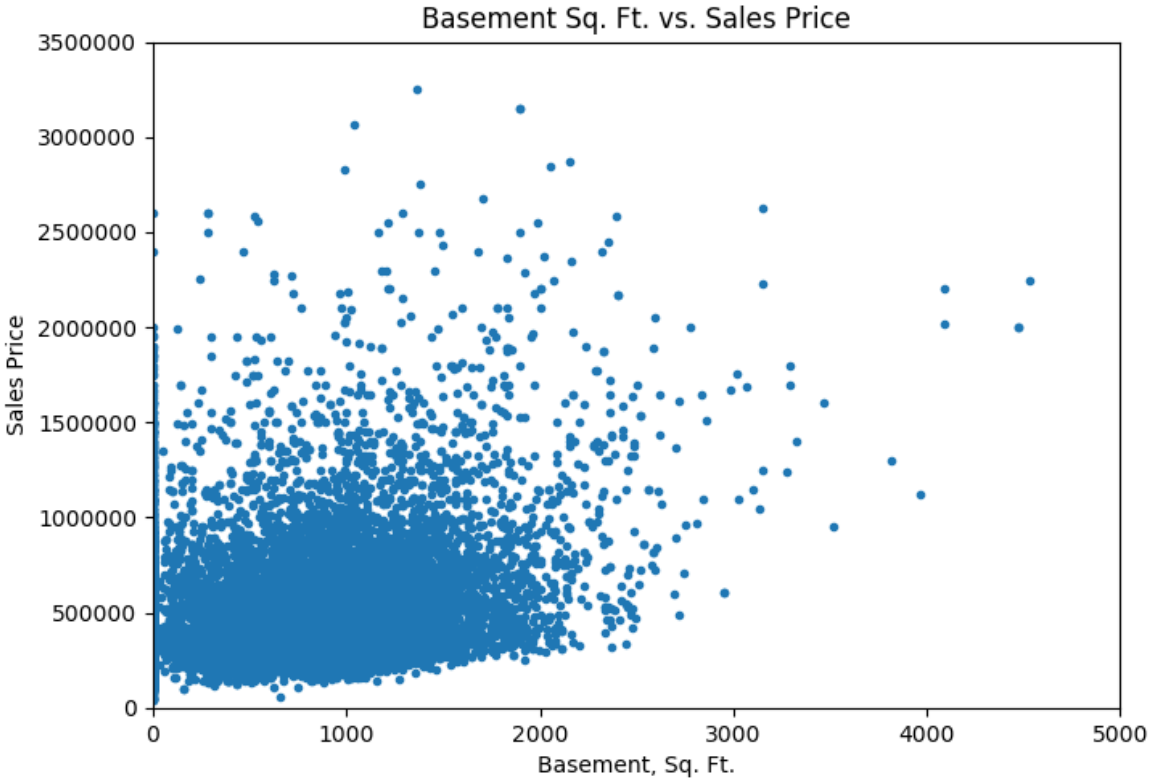

Figure 6. Basement Sq. Ft. vs Sales Price

The data gives the lot size that a home sits on. A graph of lot size vs. sales price is shown in figure 7. It appears to show less potential than the home's finished interior size, but will be kept in the model for now. It might be that large lot sizes are found mostly in rural areas, coded with a city code of "unincorporated" and that smaller lots are found in cities. Hence the lot size may have predictive value in conjunction with where the home is located.

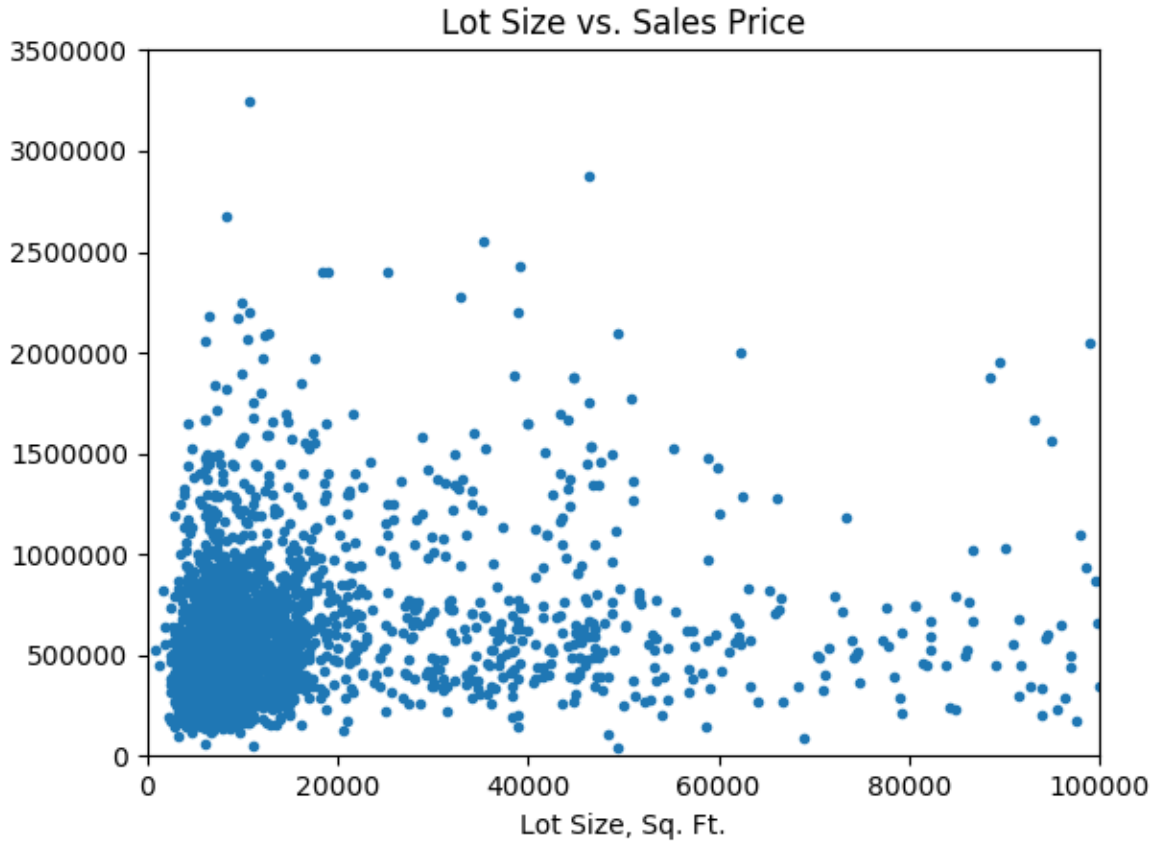

Figure 7. Lot Size vs. Sales Price.

The total number of rooms that are not bathrooms is also given in the data. There are homes with up to 20 such rooms (one outlier with over 80 rooms was removed). There are also homes coded with zero rooms. This appears to represent missing data. A box plot made in R (see figure 8) appears to show that homes with zero rooms might actually have many rooms, approximately matching the plot for as may as 14 rooms. When 0 was replaced by 14 in a linear regression model there was practically no difference in the model results, so the zeros were left in place. The box plot aggregates together data for homes with more than 15 rooms because these are relatively rare.

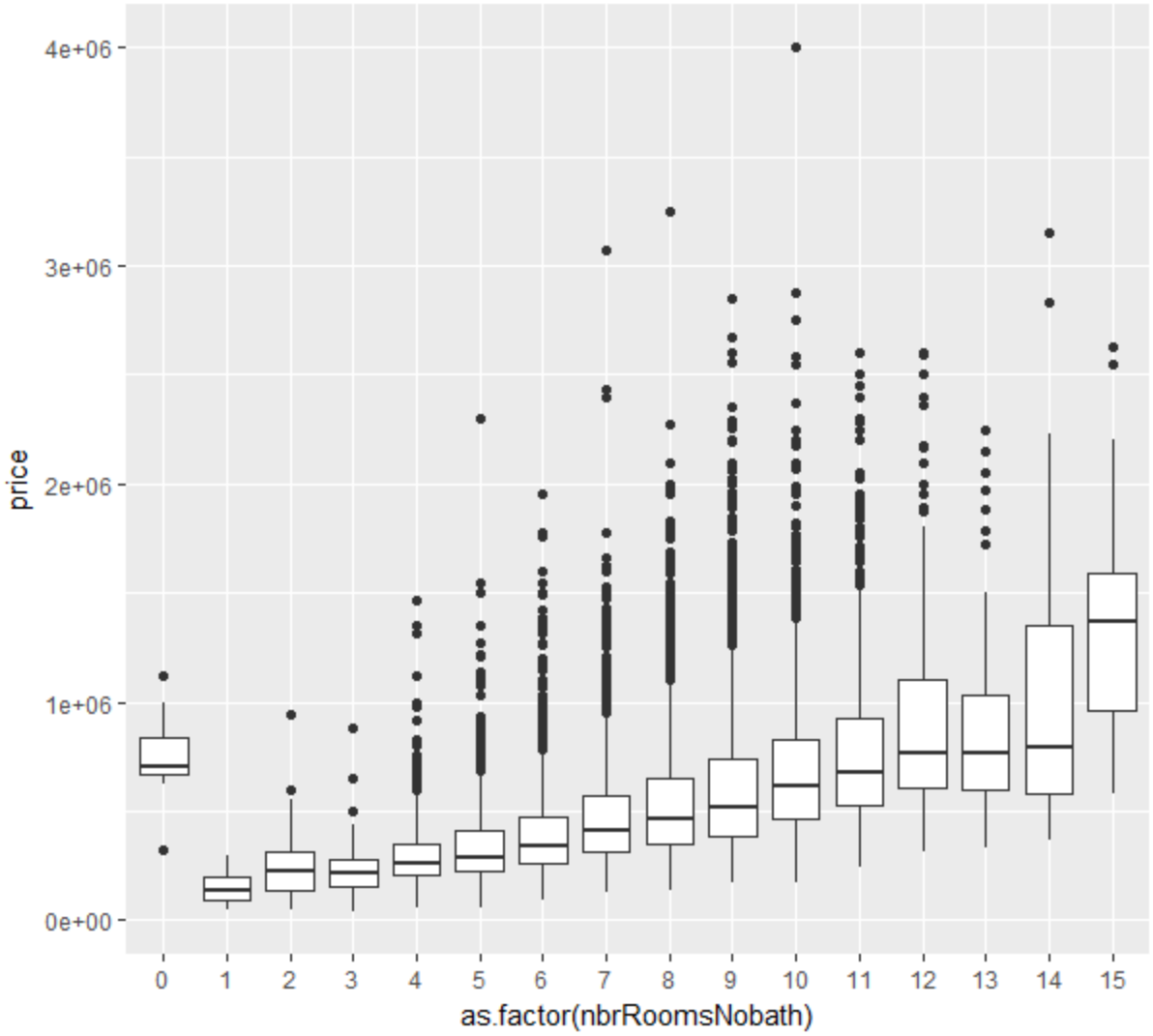

Figure 8. Number of Rooms without a Bathroom vs. Home Price

A home's age is another feature that may affect value, but the relationship is not clear from a graph of age vs. price (see figure 9). The prices range from $40,000 to $4 million. For homes less that 50 years old the plot does appear to show an increase in the minimum prices for

younger homes. Perhaps a clearer plot is a line plot of age vs. median sales price, this seems random but there may be patterns depending upon which cities and areas were built out at different times (see figure 10.

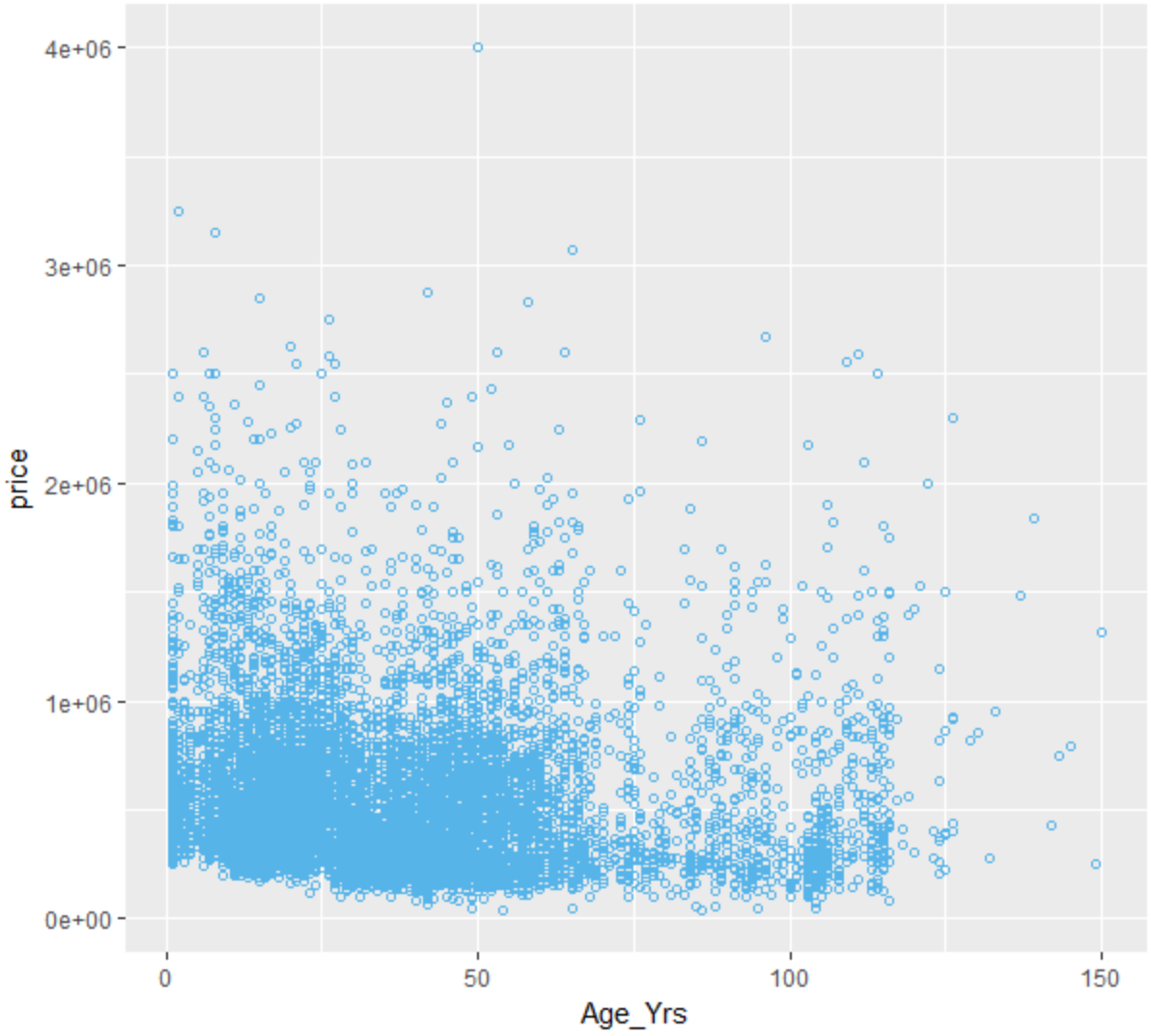

Figure 9. Home Age in Years vs. Price

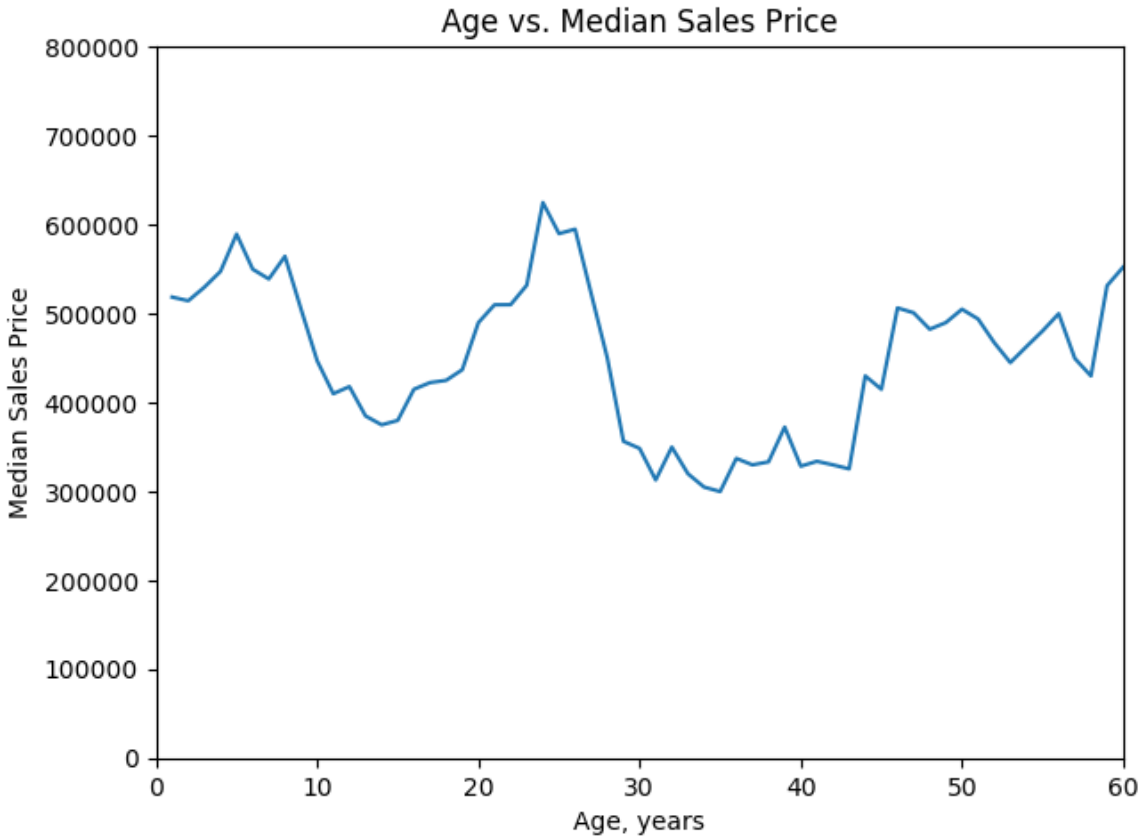

Figure 10. Home Age vs. Median Sales Price

Interestingly the effective age of a home shows more of a pattern (see figure 11). The Assessor's office defines effective in terms of either the age of the home, or the age of the last significant remodel, whichever is newer (Boulder County Assessor Public Dataset). This chart tends to show a downward trend with increasing effective age of the home. Thus newer homes and homes with newer remodels tend to have higher median prices, at least out to about 40 years old. In both of these age vs. price charts there is a local peak around 25 years old. The cause of

this peak is unknown at present, a check of age vs home size (total finished square footage) did not reveal an pattern to explain the local price peak near 25 years. Other factors such as the city that the homes were built in may be at work here.

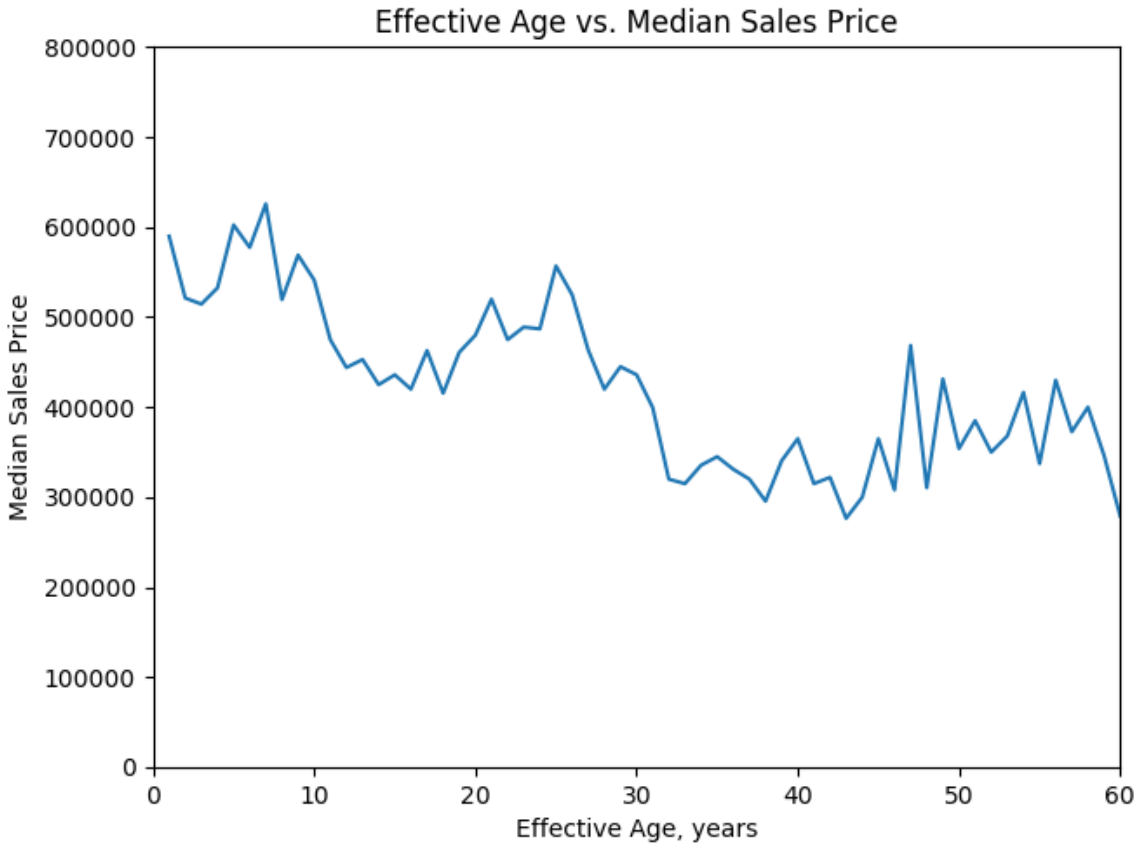

Figure 11. Effective Age vs. Median Sales Price

Metrics:

A number of different metrics were utilized in order to determine which model fits the data the best. The root mean squared error (RMSE) of the log of housing prices was used as the

primary metric (Srivastava, 2016). Kaggle states that taking the log and using this metric reduces the effect that high priced homes have on the result (House Prices: Advanced Regression Techniques, 2017). Kaggle points to a Wikipedia article to explain the metric in more detail (Root-mean-square deviation, 2017). Python supports this metric in several ways. First, the cross validation function, "cross_val_score" from the Scikit-Learn model selection library provides a metric for each data point called "neg_mean_squared_error" (sklearn.model_selection.cross_val_score). The value produced can be negated, the square root taken and the mean value calculated in order to produce the RMSE as used by Kaggle.

Because this is a regression problem, it appears that standard regression metrics between predicted and actual sales price would also be of benefit. I chose to record the R-squared, R-squared adjusted and gradient values for each model. R-squared and R-squared adjusted are standard measures in statistics for the fit of a regression. Since predicted and actual price would be identical in an ideal world, the closer this value is to one the better.

Another metric chosen to test the models against is the mean absolute percent error (MAPE). MAPE is calculated simply by calculating the percentage of error for every observation and then taking the mean for all of the data in the result. As will be shown, this metric is especially valuable for making clear the improvement in predictive power for the models that use the home feature data in addition to the assessor's estimate. The MAPE value was halved by the machine learning as compared to simply using the assessors estimate as a predictor for the actual price that a home would sell for.

Model:

Because of the the poor gradient of the assessor's estimate of value (the 'totalActualVal'

feature), and potential non-linearity in other features such as the effective age of the home, I expected that tree based models might tend to fit the data better than linear regression techniques . This is because linear regression techniques do not model non-linear relationships well (Modern Machine Learning Algorithms: Strengths and Weaknesses, 2017). Thus, several tree-based modeling methods were used to analyze the data including Random Forests, Extra Trees, Gradient Boosting and XG Boost. I also tried several linear regression models to verify my assumptions. Additionally, because of its importance, the assessor's estimate of value was squared in a linear regression model in order to test the possible benefits of a higher order polynomial model. Building a polynomial model is an area for further research, perhaps in the second practicum, time and computing resource limitations did not permit further research into this area in this first practicum.

Training data consisted of 70% of the available data. The test set consisted of the remaining 30% of the data. All scores and metrics are reported for the test data except otherwise noted. The cross validation score is done on the training data since that metric uses only the training data and makes splits into subsets for training and test within the cross validation algorithm.

Model results are shown in figure 12. “CV RMSE” is the root mean squared error for the cross validation performed on the training data. “30% RMSE” is the root mean squared error for the test set, which consists of 30% of the overall data. R-squared and R-squared adjusted are these statistical metrics for the prediction from the 30% test set vs. the actual sales prices for those homes. The gradient and intercept is calculated by Python from the 30% test set vs. the actual sales prices.

The first result row (see figure 12) is simply regression based only upon the assessor's

estimate of the home's value. This estimate suffers from poor mean absolute percent error of around 16%. The r-squared and r-squared adjusted are good, but beaten by the machine learning models that used all of the available home data. Linear regression is simply a linear regression performed on the data set using all of the available home data (assessor's estimate plus home feature data). "Linear regression w/ squared value" is a linear regression as above but with the addition of the assessor's estimate being squared to model possible non-linearities in this important variable. Lasso and Bayesian Ridge are variants of linear regression, neither performed as well as standard linear regression.

Tree type models tended to fit the data best. These included Random Forest, Extra Trees, Gradient Boosting and XG Boost. Results for these are shown in figure 12, the parameter tuning for these models is shown in figure 13. Linear regression models did not fit the data as well. The best model was a blend of several different tree type models, Gradient Boosting, Extra Trees and XG Boost. The blending was accomplished by averaging the predictions of the three different models. I also tried Support Vector Machines and a Multi-Level Perceptron, neither gave good results and though they might have benefited from tuning, due to poor initial performance they were dropped due to semester time constraints.

Based upon the tuning that could be done in the semester time frame, it appears that a Gradient Boosting model performed best of the individual models tried (see figure 12). It had the lowest RMSE values and the highest R-squared, R-squared adjusted of the single method models. The gradient boosting model also had the lowest mean absolute percent error for any of the single models tried. Its gradient was the closest to one for the tree model types, though regression was best in this area. These metrics put together means that the Gradient Boosting model produce the least error compared to all of the other single learning models.

The second best individual model was the XG Boost Regressor model. This performed almost as well as the Gradient Boosting model (see figure 12). Both models were tuned to this specific problem, the Gradient Boosting Regressor by a grid search and the XG Boost by one at a time parameter tuning. There was not time to do an exhaustive grid search on the XG Boost algorithm because of the many parameters required for tuning it and the time and computing resources available. Also preforming well was an Extra Trees model, it was only slightly less powerful that the Gradient Boost or XG Boost models and it was good enough to be helpful in the final ensemble.

As noted above, averaging ensembles (blended models) worked the best of the models tried during this project. These models work by averaging the results of two or more machine learning algorithms (Demir, 2010) (Hearty, 2016). Looking at the best blended model (Blended Model (3) in figure 12) and comparing it to simply using the assessor's estimate of value, we can see a notable increase in the predictive capability of the blended model as compared to the assessor's estimate. R-squared adjusted has increased a small amount, but more importantly the gradient and MAPE (mean absolute percent error) values have improved significantly. MAPE for the best blended model is less than half of what it was with just the assessor's estimate.

Also, we can compare the graph for using just the assessor's estimate (see figure 3) with a similar graph for the final blended model (see figure 14). In addition to less scatter of the prices from the ideal regression line (green), we can also observe that the actual regression lines (orange) are closer to the ideal regression line in the final blended model. In addition to the metrics discussed above, the graph is further evidence that the blended machine learning model has gained significant accuracy over simply using the assessor's estimate alone to predict home sales prices.

Models Comparison

Model	CV RMSE	30% RMSE	R-squared	R-squared adj	gradient	MAPE
Regression (Assessor's Estimate)	---	---	0.8937	0.8937	1.1296	16.40
Linear Regression	0.165745	0.156907	0.8551	0.8516	0.8529	11.49
Linear Regression w/ squared value	0.150515	0.141076	0.8887	0.8860	0.9655	10.30
Lasso	0.187745	0.177409	0.7956	0.7907	0.7200	13.12
Bayesian Ridge	0.165652	0.156870	0.8547	0.8512	0.8509	11.48
Random Forest	0.132738	0.125630	0.9170	0.9150	1.0117	8.83
Gradient Boosting Regressor	0.124745	0.118781	0.9254	0.9236	1.0003	8.42
Extra Trees Regressor	0.129647	0.122686	0.9208	0.9189	1.0112	8.47
XG Boost Regressor	0.126890	0.119441	0.9199	0.9180	1.0149	8.53
Blended Model (4) RF, GB, XT, XGB	---	0.116910	0.9261	0.9244	1.0167	8.20
Blended Model (3) GB, XT, XGB	---	0.115938	0.9273	0.9256	1.0159	8.14
Blended Model (2) GB, XGB	---	0.116891	0.9258	0.9240	1.0119	8.32

Figure 12. Table of Results for Various Models

```
reg1 = RandomForestRegressor(n_estimators=1000, n_jobs=3, random_state=42)
reg2 = GradientBoostingRegressor(learning_rate=0.02,
                                 n_estimators=2000,
                                 max_depth=6,
                                 subsample=0.6,
                                 max_features='auto',
                                 random_state=42)
reg3 = ExtraTreesRegressor(n_estimators=1000, n_jobs=3)
reg4 = xgb.XGBRegressor(learning_rate=0.075,
                        max_depth=6,
                        min_child_weight=1.0,
                        subsample=1.0,
                        colsample_bytree=0.4,
                        colsample_bylevel=0.8,
                        reg_lambda=1.0,
                        reg_alpha=0,
                        gamma=0,
                        n_estimators=700,
                        seed=42,
                        silent=1)
```

Figure 13. Tree Model Tuning from Python Script

Figure 14. Final Model Fit, Blended Model (Extra Trees, Gradient Boosting, XG Boost)

Project Notes:

Some of the exploratory data analysis was done in R. The R file is stored in the GitHub repository as "EDA.R" (GitHub). The majority of the programming work was done in Python, including all of the machine learning. A list of scripts may be found in the list of scripts at the end of this report. The graphs of figures 8 and 9 were produced with R, the remaining graphs in Python.

Categorical Variables:

The model uses a number of categorical variables. These are converted to dummy Boolean variables in standard practice where each categorical variable is expanded to many Boolean features. When using dummy Booleans to represent categories in a categorical variable it is normal to "leave one out" since the logical structure of the categorical variable is theoretically completely described by n-1 dummy variables (Lantz, 2013, pg 195).

However, it was found through experimentation that leaving one out actually hurt model accuracy slightly, at least with the tree-based algorithms that performed best. While this result was not entirely expected, it has been reported before with similar housing data. Choudhary reported a similar result in a script presented as part of the Kaggle housing competition (Choudhary, 2016). Hence the models reported above were built with all categories represented as Boolean features, leaving one out was not done on the data.

The results can be seen in figure 15, in all four cases studied leaving in all of the dummy features outperformed leaving one out. The metric used here is RMSE (root mean squared error) on the log of house price as discussed earlier. The lower the metric, the better the model and the higher the prediction accuracy. In all of the cases below the RMSE is lower for leaving in all of the dummy variables. Because of the results of Choudhary, this was not an unexpected result for this model. Apparently, tree based models perform better with all of the dummy coded features left in the model, perhaps because they do not necessarily consider every feature at every split in order to randomize the model trees. Researching the 'why' of this result is another possible project, or part of a project, for a second practicum.

Leave One Out Comparison				
Model	RMSE, All In	RMSE, Leave One Out	Difference	Better
Gradient Boosting Regressor	0.118781	0.119458	-0.000677	All In
Extra Trees Regressor	0.122686	0.123981	-0.001295	All In
XG Boost Regressor	0.119441	0.120598	-0.001157	All In
Blended Model (3) GB, XT, XGB	0.115938	0.116900	-0.000962	All In

Figure 15. Comparison of All Dummy Variables In vs. Leave One Out.

Conclusions:

Adding in the home's features in the assessor's data base improved the model by more than halving the mean absolute percentage error over using simply the assessor's estimate of a home's value. This resulted in a model with 100 columns of features rather than a single input feature, but the gain in accuracy appears to make the additional model complexity worth it. It should be noted that the number of features grew large because many of the categorical features needed to be converted to dummy variables where multiple Booleans represented the categories.

A blended model consisting of a Gradient Boosting Model, an Extra Trees Model and an XG Boost model performed best at improving the estimate of a home's sales price as compared to simply using the assessor's estimated value. The blended models, blended by averaging the results of multiple tree based models, outperformed any of the individual machine learning models attempted for this project. If we compare the graph of figure 3 using just the assessor's estimate with the final machine learning model result in figure 14, we can notice that the regression line lies much closer to the ideal regression line and that there is less scatter of homes from the regression lines. Adding in home features and machine learning has allowed for better predictions to be made of the sales prices of homes in Boulder county then was possible by using the assessor's estimate alone.

References:

Boulder County Assessor Public Dataset. PDF file obtained from the assessor's office along with the data. Available from: https://github.com/Vettejeep/Boulder_County_Home_Prices.

GitHub. Git Hub repository for the project. Retrieve from: https://github.com/Vettejeep/Boulder_County_Home_Prices.

House Prices: Advanced Regression Techniques (2017). Retrieved from: https://www.kaggle.com/c/house-prices-advanced-regression-techniques#evaluation.

Root-mean-square deviation. Wikipedia (2017). Last edited on 1 June 2017, at 03:57. Retrieved from: https://en.wikipedia.org/wiki/Root-mean-square_deviation.

sklearn.model_selection.cross_val_score (2016). Scikit-Learn. Retrieved from: http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html.

Modern Machine Learning Algorithms: Strengths and Weaknesses (2017). Elite Data Science. Retrieved from: https://elitedatascience.com/machine-learning-algorithms.

Srivastava, Tavish (2016). 7 Important Model Evaluation Error Metrics Everyone should know. AnalyticsVidhya. Retrieved from: https://www.analyticsvidhya.com/blog/2016/02/7-important-model-evaluation-error-metrics/.

Hearty, John (2016). Advanced Machine Learning with Python. Packt. Birmingham, U.K.

Demir, Necati (2010). Ensemble Methods: Elegant Techniques to Produce Improved Machine Learning Results. Toptal. Retrieved from: https://www.toptal.com/machine-learning/ensemble-methods-machine-learning.

Lantz, Brett (2013). Machine Learning with R. Packt. Birmingham, U.K.

Choudhary, Amit (2016). script_v6. Kaggle. Retrieved from: https://www.kaggle.com/amitchoudhary/script-v6.

Scripts:

Assemble_Data.py: Data assembly, table joins, feature selection and processing of dummy variables

EDA.R: Exploratory data analysis in R.

blended_tree_models.py: Obsolete first effort at model blending

plots.py: Exploratory data analysis, plotting in Python

sales_clean.py: Clean up for the sales data file, extracts year and month data, real property and sales code indicating arms length transactions

single_model_regression.py: linear regression models

tree_models.py: machine learning tree models

tree_models_four.py: 3-4 tree models blended, the final model was produced from this file

value_vs_price.py: analysis using just the assessor's estimate of value

xl_to_csv: convienence script, converts “xlsx” files to csv, made it easier for OpenOffice to read them.