Modeling NYC Real Estate Property Sale Price

Ethan Anderson, Jacob Malovich, Zion Steiner, Heber Jenson

INTRODUCTION

Real estate is a massively valuable market in most parts of the world. In the U.S. alone, the cumulative price of homes is estimated to be worth $31.8 trillion[1]. Because there is a limited amount of land, real estate is a largely speculative market. Eventually, all useful land will be owned, and the value of land can only increase. Investors buy and sell properties based on their expected future value, which determines the expected return on investment. Property prices are related to the quality of the land and building construction, but location also plays a large role. For instance, city properties tend to be the most expensive because of the close proximity to public services, entertainment venues, job opportunities, and generally high demand.

Because there is a lot of money to be made or lost in the real estate market, it is valuable to be able to accurately predict property sale price. There are many variables that determine property value, so an analysis needs to be done to select the most informative of these. These can then be used in the development of an OLS (ordinary least squares) model.

A model that can effectively predict property prices is useful for several applications. Knowing the estimated value of a hypothetical development can be useful to both prospective real estate firms and those in the market for housing. For example, a firm can use the model to predict the estimated sale price of a prospective building project. Firms can use this figure to adjust the location, size, and the number of units to reach a given profit from a project. Similarly, those in the market for a house can use property price predictions to find the "fair" price for houses they are interested in. If a home they are looking at is being sold above the predicted price, they can know that it is overvalued. The homebuyers could then use this information to negotiate the price with the seller, or just cross it off their list altogether.

In this paper, we describe the construction of a model that can predict residential property sale price in New York City. We start out with an initial OLS model composed of all available predictive features and use a variety of techniques to improve it. We then discuss the effectiveness of the final model in comparison with more powerful modeling approaches including regression trees, and a brief comparison with random forest regression.

DATA

The data we will be using comes from the New York City Department of Finance's Rolling Sales dataset. The initial dataset included over 80,000 sales or transfers of property in the city of New

[1] https://www.barrons.com/press-release/PR-CO-20171228-906128?tesla=y

York. However for the purpose of data analysis the data needed some initial manipulations. For example, listed in the dataset were transfers of property ownership either through wills, or from parents to children, resulting in low values such as 0. The initial dataset also included a few outliers at really high values over $5 million. Removal of these influences trimmed down the dataset to a more reasonable range. A QQ-Plot of sale price is shown below to show the range of the cleaned data set.

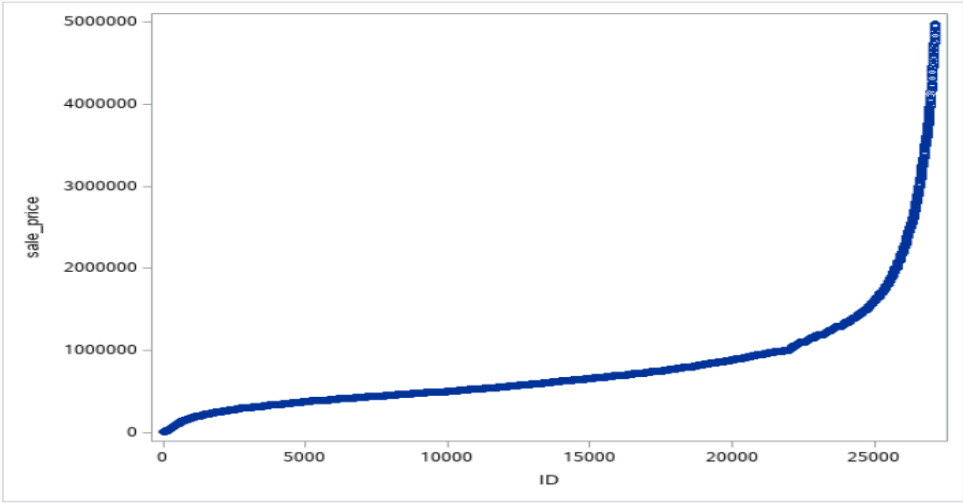


Figure 1 Sale price ranges from low $1000's to $5 million

The data included 22 variables which detailed each property, including the Tax Class, Total Units, Residential or Commercial, Borough, Neighborhood, Easement, Sales Price, Address, Year Built, and so on. Some of these variables were excessive in the presence of other variables and were not practical to include in the analysis. The variables that were used in the analysis made sense intrinsically. Some of these variables include Gross Square Feet, Number of Residential or Commercial Units, Tax Class at Time of Sale, Sale Year, and Year Built. Two additional changes were made to the initial variables in that we replaced Year Built into Building Age so that the variable would be easier to interpret, and we also use the Sale Date to create a Sale Season variable, representing quarters of the year, to see if there is also a correlation between price and sale quarter. As people have a tendency to move during the warmer seasons of the year.

With additional research, we wanted to include in the model the possible effect crime has on housing price. We decided to add the crime statistics for 2015 associated with each zip code included in the data set. This data was not found in the original observations obtained from the New York City Department of Finance's Rolling Sales Dataset. This data was collected from the NYPD complaint database[2], which records all crime reports with coordinates. Crime count was

[2] https://data.cityofnewyork.us/Public-Safety/NYPD-Complaint-Data-Historic/qgea-i56i

aggregated by zip code and then joined with the zipcode of buildings in the property sales database. The rationale behind this is that 2015 would be the most recent complete year of statistics a buyer in 2016 or 2017 would have, as they considered a purchase of a new property. The Sales Price floor was set to above 10,000 to remove any outliers on the low end and a few observations with 0 in the Gross Square Feet variable value were removed. After all the changes to the dataset for the initial analysis we ended up having approximately 27,000 observations.

The variables Borough, Tax Class, Sale Year and Sale Season are qualitative variables, only taking on a specific set of values. Borough represents the district or town the property is found in. In the dataset there are 5 different boroughs including Brooklyn, Queens, Manhattan, Bronx, and Staten Island. Tax class takes the value of 1, 2, or 4 to represent the type of class it is. 1 being a property with up to 3 units, residential or commercial. 2 represents rental properties, condominiums, and co-ops. Class 4 is all other properties not in those classes. There were no observations in the dataset for tax class 3 as it is representative of utilities buildings or structures. Sale year takes on the value of 2016 or 2017, and Sale Season takes on a value of 1 to 4, with each value representing a quarter of the year.

MODEL CONSTRUCTION

Initial Model

In order to determine what model would be most practical to use for this data set we began by testing to see if the data would meet the required parameters for linear regression. The dataset being used however limited what tests would be effective in issuing us useful results. BF testing was ineffective due to the sheer amount of data being used. Any variance in the data would cause the test results to show issues of heteroskedasticity. In order to move forward with testing, we needed to rely on visual testing to verify the data met linear regression assumptions.
Testing began with a crude OLS model containing all variables from our cleaned dataset. The resulting plots show that the model residuals are not normally distributed, meaning a transformation is necessary to help normalize the distribution of the data. The variance of the residuals also seems to increase with predicted value, meaning the residuals are not homoskedastic.

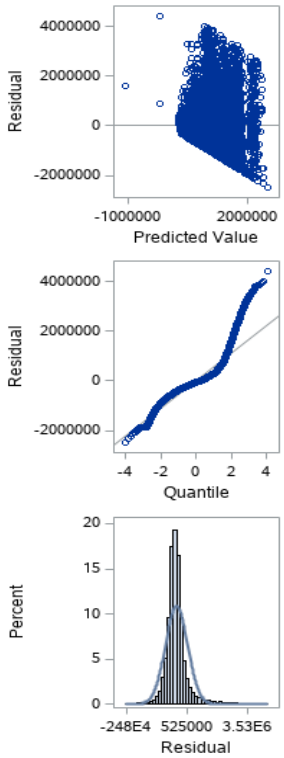


Figure 2 Heteroscedastic and not normally distributed residuals on initial model

Further analysis from the residuals by regressors for Sale_Price showed additional issues with the data. We found extremely large values contained in the variables residential units, commercial units, gross square feet, and building age. For the first three previously mentioned variables we applied a log transformation to them in order to normalize the distribution of the data. Building Age, however, had a different problem entirely. On further investigation it was found that there were four, zero values listed for the year built in the original dataset. These values caused the value for building age to be listed as 2017 matching up with the year sold. In order to remedy this issue these four observations were removed from the data set. In addition to all of the above changes we did have a couple of incredibly strong outliers and influential points. The values being outputted were influence values of 1 which is the highest it could be. In finding this point it was determined that these observations contained 5 - 20 times more commercial units than any other observation. Further analysis into commercial units showed additional findings in the data. Only 1,989 observations contained any commercial units. Of those observations the majority only contained 1 commercial unit. This led to more investigation into the data. After additional analysis into the data for the variables residential units and commercial units we determined to adjust the data one more time. We ended up removing approximately 800 observations that either contained no residential units and no commercial units, or was one of the extreme values for commercial units with a value of 100 or more. These adjustments were necessary in order to accurately report the effects that residential or commercial units have on a

buildings sale price. If the building did not contain either type of unit it didn't make sense to keep the observation.

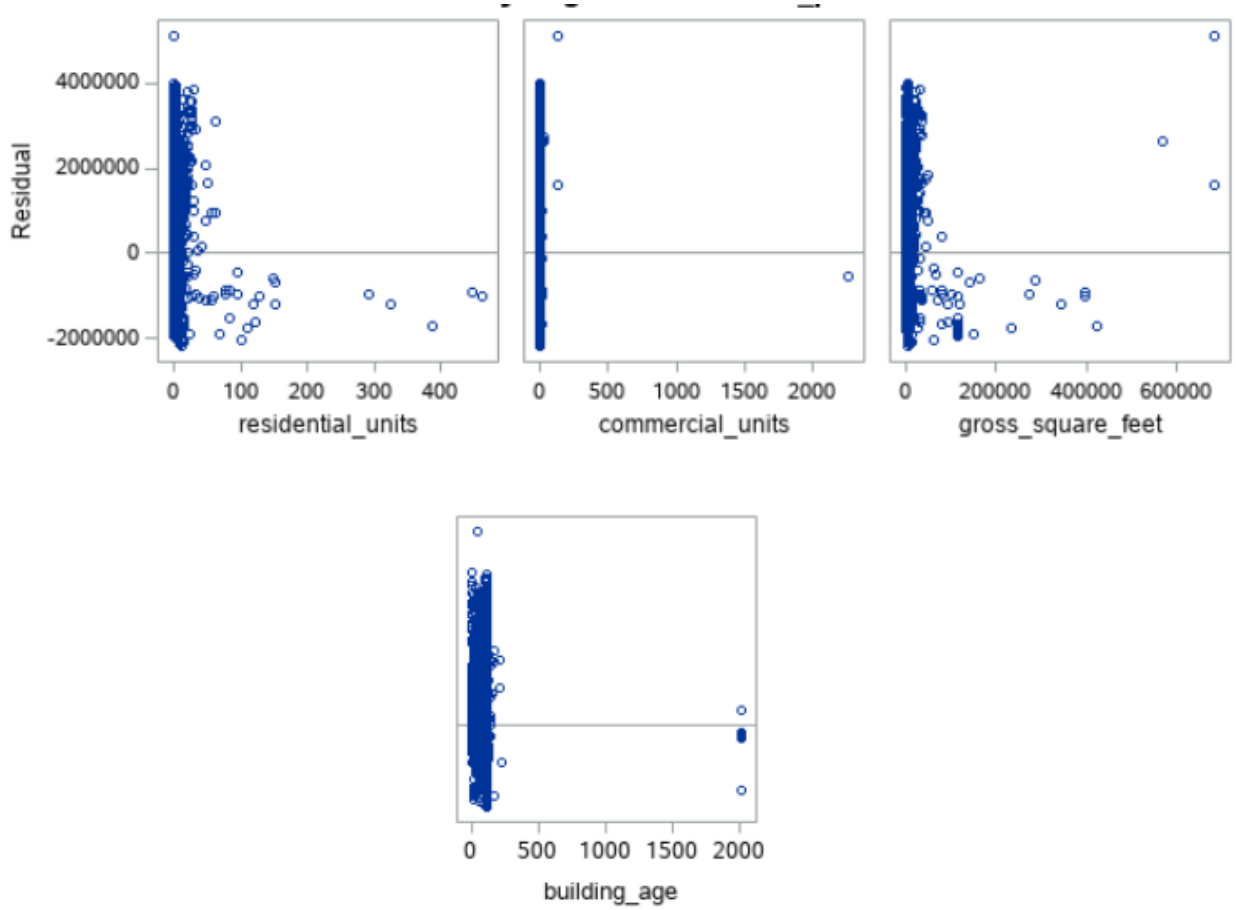


Figure 3 Residuals are not normally distributed for high feature values

We applied a log transformation to *sale_price*, *gross square footage*, the number of *commercial units*, and the number of *residential units* to try to normalize the distribution of the residuals. Below are the diagnostics from the model fit to *log(sale_price)*. The transformation reduced heteroskedasticity in the residuals, but the distribution is now left-skewed instead of right-skewed. From the visual diagnostics, our assumptions are not met. The numerical diagnostics agree. The Brown-Forsythe test of constant variance gives a p-value of 1.8372E -111, which indicates heteroskedasticity. The test of normal residuals has a correlation value of 0.9278, which is not high enough given our sample size. As mentioned previously the BF test will always be low because of the sheer amount of data being tested. Despite only our best efforts, the linear assumptions have not been met. Keeping this in mind we understand that linear regression models should not be effective in analyzing this dataset.

P-value for Brown-Forsythe test of constant variance in Residual vs. Predicted

Obs	t_BF	BF_pvalue
1	22.5649	1.8372E-111

Pearson Correlation Coefficients, N = 21702
Prob > |r| under H0: Rho=0

	resid	expectNorm
resid Residual	1.00000	0.92782 <.0001
expectNorm	0.92782 <.0001	1.00000

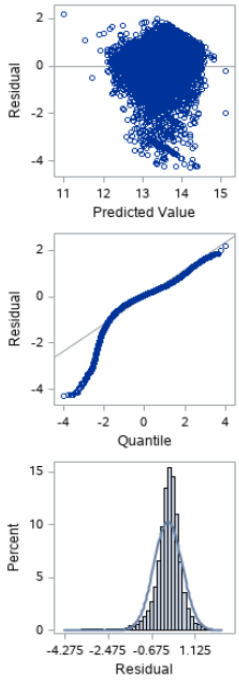


Figure 4 Residual assumptions are not fixed by initial remedial measures

Variable Selection

In order to find the most effective variables to predict log sale price we used both comparison selection as well as stepwise selection. Comparison selection used the following tests: Mallow's Cp, Akaike Information Criterion, and the Adjusted R2 in order to determine the most effective model. (See figure below for results) Comparison selection allowed us to compare a variety of tests in order to find the best variables to include in the model. Using this output we could compare it with the results from stepwise selection in order to find the best model.

Number in Model	Adjusted R-Square	R-Square	C(p)	AIC	SBC	Variables in Model
14	0.2734	0.2739	14.6759	-22372.238	-22253	Col3 Col4 Col5 Col6 Col7 Col8 Col9 Col10 Col12 Col13 Col14 Col16 Col17 Col20
14	0.2734	0.2739	14.6759	-22372.238	-22253	Col3 Col4 Col5 Col6 Col7 Col8 Col9 Col10 Col11 Col13 Col14 Col16 Col17 Col20
14	0.2734	0.2739	14.6759	-22372.238	-22253	Col3 Col4 Col5 Col6 Col7 Col8 Col9 Col11 Col12 Col13 Col14 Col16 Col17 Col20
14	0.2734	0.2739	14.6759	-22372.238	-22253	Col3 Col4 Col5 Col6 Col7 Col8 Col9 Col10 Col12 Col13 Col15 Col16 Col17 Col20
14	0.2734	0.2739	14.6759	-22372.238	-22253	Col3 Col4 Col5 Col6 Col7 Col8 Col9 Col10 Col11 Col13 Col15 Col16 Col17 Col20
14	0.2734	0.2739	14.6759	-22372.238	-22253	Col3 Col4 Col5 Col6 Col7 Col8 Col9 Col11 Col12 Col13 Col15 Col16 Col17 Col20
14	0.2734	0.2739	14.6759	-22372.238	-22253	Col2 Col3 Col5 Col6 Col7 Col8 Col9 Col10 Col12 Col13 Col14 Col16 Col17 Col20
14	0.2734	0.2739	14.6759	-22372.238	-22253	Col2 Col3 Col5 Col6 Col7 Col8 Col9 Col10 Col11 Col13 Col14 Col16 Col17 Col20
14	0.2734	0.2739	14.6759	-22372.238	-22253	Col2 Col3 Col5 Col6 Col7 Col8 Col9 Col11 Col12 Col13 Col14 Col16 Col17 Col20
14	0.2734	0.2739	14.6759	-22372.238	-22253	Col2 Col3 Col5 Col6 Col7 Col8 Col9 Col10 Col12 Col13 Col15 Col16 Col17 Col20
14	0.2734	0.2739	14.6759	-22372.238	-22253	Col2 Col3 Col5 Col6 Col7 Col8 Col9 Col10 Col11 Col13 Col15 Col16 Col17 Col20
14	0.2734	0.2739	14.6759	-22372.238	-22253	Col2 Col3 Col5 Col6 Col7 Col8 Col9 Col11 Col12 Col13 Col15 Col16 Col17 Col20

Figure 5 Somehow, many models have the exact same performance metrics

Stepwise selection was used in addition in order to effectively find the most important variables to be included in the model. The threshold for entry and staying was set to an alpha level of .05 to ensure high significance.

Parameter Estimates						
Variable	Label	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	Intercept	1	9.05053	0.09219	98.17	<.0001
Col9	log_gross_square_fee	1	0.62061	0.01274	48.71	<.0001
Col4	BOROUGH 3	1	0.45099	0.01417	31.84	<.0001
Col13	crime_count	1	-0.00002902	0.00000116	-25.00	<.0001
Col5	BOROUGH 4	1	0.14138	0.01201	11.77	<.0001
Col10	TAX_CLASS_AT_TIME_OF 1	1	-0.27566	0.02196	-12.55	<.0001
Col14	sale_year 2016	1	-0.07885	0.01716	-4.59	<.0001
Col7	log_residential_unit	1	-0.21039	0.01828	-11.51	<.0001
Col19	sale_season 4	1	0.04149	0.02059	2.02	0.0439
Col18	sale_season 3	1	0.05451	0.01346	4.05	<.0001
Col20	building_age	1	0.00089317	0.00015097	5.92	<.0001
Col16	sale_season 1	1	-0.03299	0.01150	-2.87	0.0041
Col8	log_commercial_units	1	-0.14102	0.02567	-5.49	<.0001
Col3	BOROUGH 2	1	-0.11104	0.01618	-6.86	<.0001

Figure 6 Log(sqft) is chosen first by stepwise selection

Unfortunately the two selection methods did not agree on all of the variables that should be included in the model. Comparison selection reported a model with 14 variables would be most

effective, while stepwise supported 13. In the end we choose to follow the stepwise report for the model for variable selection.

Interaction Terms

With variable selection completed we were curious to test 3 specific interactions in order to see if they would have an impact in predicting log sale price. The three interactions included crime and building age, followed by gross square feet and commercial units, and gross square feet and residential units. Crime and age were two variables that we wanted to test in order to see if the age of the building would lead to more crime. In testing all three interactions we found that only the age and crime interaction did not have a significant impact in predicting log sale price.

The final model with parameters is shown below with variance inflation factors as well as their collinearity condition index. The high values for VIF should not be an issue as they are caused due to the interaction terms.

Parameter Estimates

Variable	Label	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	Intercept	1	7.40862	0.11323	65.43	<.0001
Col9	log_gross_square_fee	1	0.81309	0.01472	55.23	<.0001
Col4	BOROUGH 3	1	0.40804	0.01417	28.79	<.0001
Col13	crime_count	1	-0.00003563	0.00000275	-12.98	<.0001
Col5	BOROUGH 4	1	0.12598	0.01201	10.49	<.0001
Col10	TAX_CLASS_AT_TIME_OF 1	1	-0.23556	0.02178	-10.82	<.0001
Col14	sale_year 2016	1	-0.08654	0.01686	-5.13	<.0001
Col7	log_residential_unit	1	1.37800	0.06526	21.11	<.0001
Col19	sale_season 4	1	0.04515	0.02022	2.23	0.0256
Col18	sale_season 3	1	0.05691	0.01322	4.30	<.0001
Col20	building_age	1	0.00057499	0.00024875	2.31	0.0208
Col16	sale_season 1	1	-0.03654	0.01130	-3.23	0.0012
Col8	log_commercial_units	1	-0.53825	0.19282	-2.79	0.0053
Col3	BOROUGH 2	1	-0.14309	0.01615	-8.86	<.0001
gross_commercial		1	0.05625	0.02265	2.48	0.0130
age_crime		1	4.773936E-8	3.031166E-8	1.57	0.1153
gross_residential		1	-0.17811	0.00726	-24.54	<.0001

Figure 7 High VIF from interaction effects

$$\widehat{log(salesPrice)} = 7.37 + 0.815 * log(sqft) + 0.40 * borough3 - 0.00003 * crimeCount + 0.122 * borough4 - 0.235 * taxClass1 - 0.086 * sold2016 + 1.37 * log(residentialUnits) + 0.045 * season4 + 0.057 * season3 + 0.00089 * buildingAge - 0.036 * season1 - 0.544 * log(commercialUnits) - 0.147 * borough2 + 0.056 * sqft * commercialUnits - 0.17 * sqft * residentialUnits$$

The interpretation of any of the log variables would be that for every one percent increase in that variable we would expect to see a change in sale price by beta percent. For example, holding all other variables constant, for every one percent change in gross square feet, we would expect to see a 0.815 percent increase in sale price. For variables that do not contain a log, we would interpret them as follows. Holding all other features constant, a one unit increase in variable x, we would expect to see a 100*beta change in sale price. Let's use borough 3 as an example. Holding all things constant, if the property is located in borough 3, we would expect to see a 40.466% increase in sale price.

Influential Points

In the beginning analysis of the data we found very strong influential points. These observations had extremely high amounts of commercial units. After removal of these data points our data is not greatly affected by influential points. The plot of the leverage points shows how our removal of the previous outliers greatly impacted the effects of the data. All current influential points have minimal leverage values, allowing us to confidently move forward with our model.

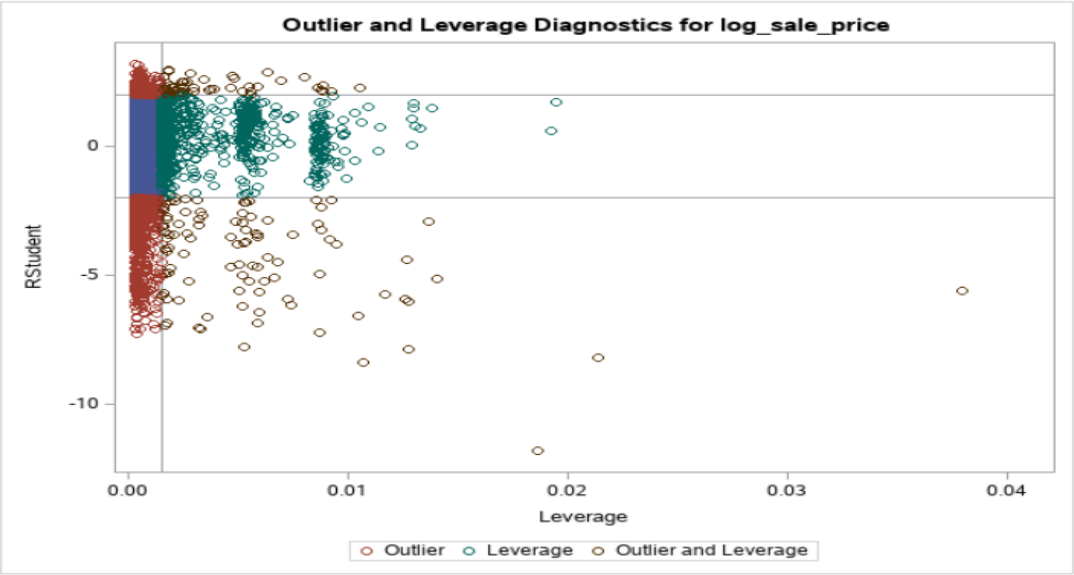


Figure 8 Remaining points have small leverage after removal of high leverage point

Model Assumptions

Unfortunately, the final linear model does not meet the assumptions of OLS, even after all the corrective measures that were applied. The residual distribution is still right-skewed. Because of this, we can conclude that OLS is not the model most suited for this problem. To test this claim, we explored using the regression trees and a random forest algorithm on this dataset to get a comparison.

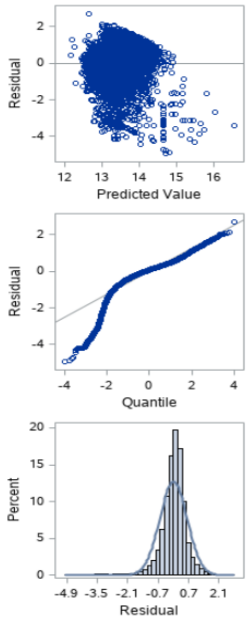


Figure 9 OLS assumptions still not met, even after remedial measures

ALTERNATIVE MODEL

- Regression Trees

We chose to use a regression tree analysis on this data as regression trees do not require the same restrictive assumptions that linear regression models do. This would allow us to analyze the data without requiring it to be normally distributed, an error that we were never able to resolve using linear regression. Using SAS we included all variables we used in the final dataset. After running the program for a few hours the computer generated the following output. The output is hard to

read as the computer had so many models to attempt to calculate. We will make our best attempt in reporting the results.

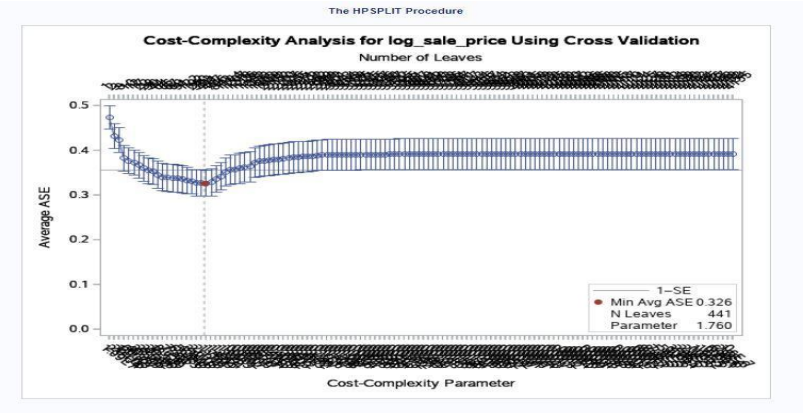
The HPSPLIT Procedure

Performance Information	
Execution Mode	Single-Machine
Number of Threads	2

Data Access Information			
Data	Engine	Role	Path
WORK.NYC	V9	Input	On Client
WORK.OUT2	V9	Output	On Client

Model Information	
Split Criterion Used	Variance
Pruning Method	Cost-Complexity
Subtree Evaluation Criterion	Cost-Complexity
Number of Branches	2
Maximum Tree Depth Requested	15
Maximum Tree Depth Achieved	15
Tree Depth	15
Number of Leaves Before Pruning	4336
Number of Leaves After Pruning	435

Number of Observations Read	26277
Number of Observations Used	26277



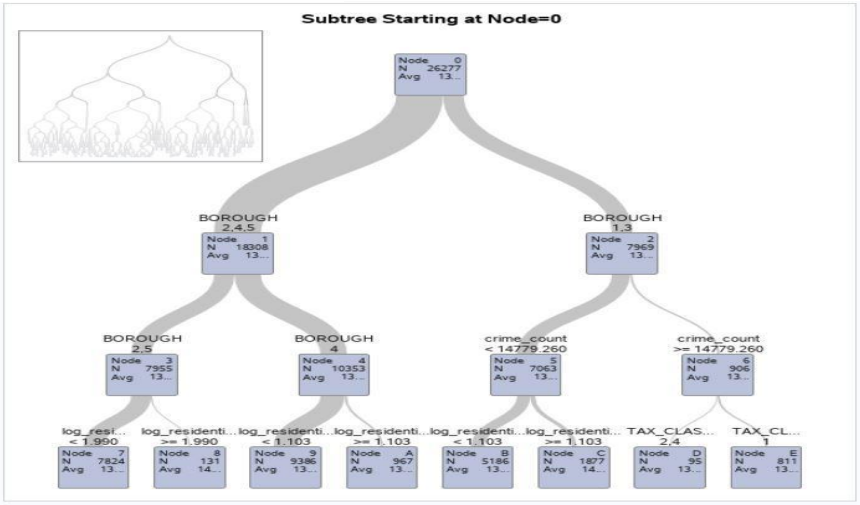


Figure 10 Partial visualization of final tree

According to the output the model created a full grown tree with a total number of leaves of 4,336. After pruning the model the tree included 441 leaves. It can be seen above that the image does not represent the entire tree. However, we were able to use this model in an attempt to predict the log sale price on a test set of the data. We calculated the MSPR to be 0.492216. This number will be used as a method for comparison against our linear model.

-Random forest regression in Python

Because the data did not follow the model assumptions of OLS we had to use a different regression technique to analyze our data. For SAS application, we chose to use a Regression Tree approach. Random Forest was another method we wanted to use in our analysis. We felt confident in our abilities to use python in order to run this model. We set the number of decision trees for random forest to 100 and it returned a R2 value of 0.91. In addition we created a feature importance chart reporting the importance of each variable. From this output we can see that the most important variable is log gross square feet. This makes sense as a property with more square feet would be asking for a higher price compared to one with less total square feet.

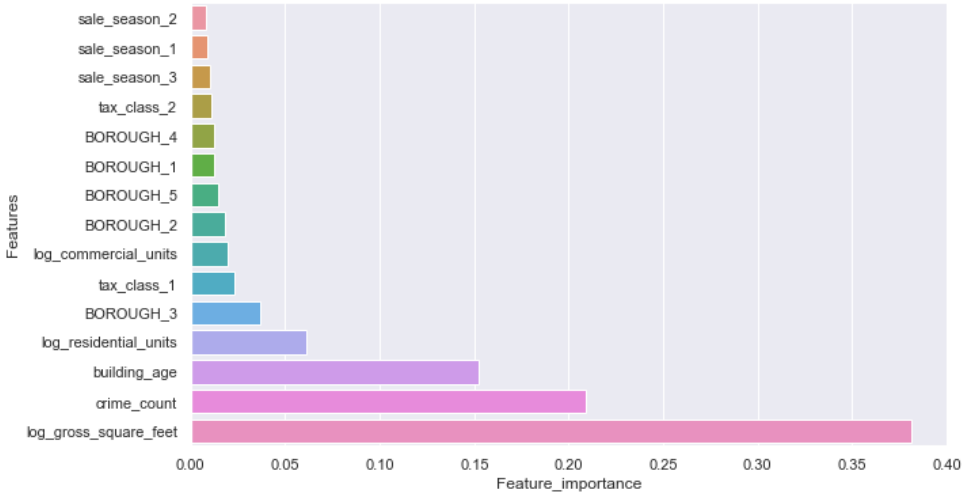


Figure 11 Feature importance per sklearn's random forest built-in method

Accuracy

In order to accurately compare the models we calculated the MSPR for each model. As a standard of comparison we also calculated the MSPR for the average of the log of sale price. This value tells us if our models make any improvements in predicting the log sale price of a property than just using the average of the training-set data.

MSR for test set using Average run

The MEANS Procedure

Analysis Variable : SqPredError
Mean
0.4754544

Figure 12 MSPE for fit intercept

MSPR for test set

The MEANS Procedure

Analysis Variable : MSE				
N	Mean	Std Dev	Minimum	Maximum
5256	0.3418940	1.1544871	2.12559E-11	21.6429279

Figure 13 MSPE for OLS

The MEANS Procedure

Analysis Variable : ASE				
N	Mean	Std Dev	Minimum	Maximum
5256	0.4922160	1.2070023	8.7777679E-9	16.6695701

Figure 14 MSPE for decision tree

Comparing the results of the MSPR shows a rather interesting result. Using the average of the training-set data the MSPR value was 0.475. It would be anticipated that all models would do better than the average, however, the regression tree model did worse. It is believed that this model is still overfitting the data from the training-set. Further analysis should be done in order to correct any error in the code. The random forest model we created in Python reported a MSPR value of 0.291. Based on the MSRP values of each model, the random forest model is the most accurate.

CONCLUSION

In this paper, we constructed two models to predict NYC property sales price. The first model used was OLS. We first applied log transformations to several of the variables and the target (sales price) to remedy heteroscedasticity of residuals, but this did not solve the problem. We also tried removing an influential point that might have hindered model performance, but its removal did not drastically change model performance. The variables that created the model with the highest adjusted R2 value was chosen as the final model. The result still fit the data poorly and violated the assumptions of OLS.

A regression tree model was used next in an attempt to increase our ability to predict log sale price of real estate in the New York Area. All variables were added to the SAS code and the model was set to investigate to a depth of 15. This process took almost 3 hours to run and generated the final model. Using this model to predict log sale price of the test set, we were able to record a MSPR value of 0.492. This was not an improvement to either linear regression or using the average of the log sale price. It was not anticipated to get these results as it was believed that regression trees would be a better model to use for this dataset. It would be our hope to continue to investigate why this model did so poorly.

The third model was random forest, it performed much better than the OLS model. The results from this research show that random forest can predict property log sales price in NYC better than OLS. In addition, random forests can handle complex non-linear variable interactions, which is why random forest has more success on this dataset. The MSPR for the random forest was 0.291, while the MSPR for OLS was only .342. Using a feature importance plot we identified square footage, crime activity, and building age as the most important features for predicting sales price for the random forest model.

We see great potential for the practical use of statistical modeling for property sales. It can be used to give investors and home searchers valuable insight into typical prices given specific property criteria. In the future, this might be implemented into an intelligent home appraisal service for providing users with useful pricing estimates. Future work toward this includes identifying the best monetization scheme for this appraisal service and adding more variables that are predictive of sales price to the dataset is also an area for potential model improvement. Additional variables to include in further tests would be distance to public transportation, ratings of local schools, and number of parks in zip code. These variables could lead to greater ability to predict the sale price of properties in the future.

APPENDIX - Python Code

```
# Load in data
import pandas as pd
data = pd.read_csv('sales-crime.csv')
data.head()
data.info()

# remove useless columns
data.drop(['Unnamed: 0'], axis = 1, inplace = True)
data.drop(['Unnamed: 0.1'], axis = 1, inplace = True)
data.drop(['Unnamed: 0.1.1'], axis = 1, inplace = True)

# construct date time variable
data['SALE DATE']= pd.to_datetime(data['SALE DATE'],
errors='coerce')
data['sale_year'] = pd.DatetimeIndex(data['SALE DATE']).year
data['sale_month'] = pd.DatetimeIndex(data['SALE DATE']).month
pd.crosstab(data['sale_month'],data['sale_year'])

# Count blanks and null values
data.replace(' ',np.nan, inplace=True)
data.isna().sum()

# Remove not needed variables
data.drop(['EASE-MENT', 'APARTMENT NUMBER', 'NEIGHBORHOOD',
'BUILDING CLASS CATEGORY', 'TAX CLASS AT PRESENT', 'BLOCK',
'LOT', 'BUILDING CLASS AT PRESENT', 'ADDRESS', 'ZIP CODE',
'TOTAL UNITS', 'LAND SQUARE FEET', 'BUILDING CLASS AT TIME OF
SALE', 'SALE DATE'], axis = 1, inplace = True)

# Remove prices below 10,000 and above 5,000,000
data = data[(data['SALE PRICE'] > 10000) & (data['SALE PRICE'] <
5000000)]
data = data[(data['GROSS SQUARE FEET'] > 0)]


# Create variable sale season
data['sale_season'] = 0
data.loc[(data['sale_month'] >= 1) & (data['sale_month'] <= 3),
'sale_season'] = 1
```

```
data.loc[(data['sale_month'] >= 4) & (data['sale_month'] <= 6),
'sale_season'] = 2
data.loc[(data['sale_month'] >= 7) & (data['sale_month'] <= 9),
'sale_season'] = 3
data.loc[(data['sale_month'] >= 10) & (data['sale_month'] <=
12), 'sale_season'] = 4

# Create variable building age
data['building_age'] = 0
data['building_age'] = data['sale_year'] - data['YEAR BUILT']

# Remove sale_month and YEAR BUILT
data.drop(['sale_month', 'YEAR BUILT'], axis = 1, inplace =
True)

# Save dataframe as csv
data.to_csv('NYC-Property-Sales-Clean.csv', index = False)
```

Python random forest

```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
sns.set()
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression

data = pd.read_csv('NYC-Property-Sales-Clean.csv')

# get dummy variables
data = pd.get_dummies(data = data, columns = ['sale_season',
'BOROUGH', 'sale_year'])
data = pd.get_dummies(data = data, columns = ['TAX CLASS AT TIME
OF SALE'], prefix = 'tax_class')

#perform log transformations
data['log_sale_price'] = np.log(data['SALE PRICE'])
data['log_gross_square_feet'] = np.log(data['GROSS SQUARE
FEET'])
```

```
data['log_residential_units'] = np.log(data['RESIDENTIAL UNITS']
+ 1)
data['log_commercial_units'] = np.log(data['COMMERCIAL UNITS'] +
1)

# create test set and training set
y = data['log_sale_price']
x = data.drop('log_sale_price', axis = 1)

x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.2, random_state = 12345)

# fit random forest regression
RF_regressor = RandomForestRegressor(n_estimators = 100,
max_features='sqrt', random_state = 12345)
RF_regressor.fit(x_train, y_train)

# calculate predictions
y_pred_test = RF_regressor.predict(x_test)


# calculate MSPR
MSPR_test = round(np.mean(np.square(y_test - y_pred_test)),6)
print("MSPR:", MSPR_test)

# calculate r-squared
r_squared = RF_regressor.score(x_train, y_train)
print("R-squared:", round(r_squared, 3))

# feature importance
features= list(x_train.columns)
importances= RF_regressor.feature_importances_
FIM = pd.DataFrame({'Features': features ,
'Feature_importance':importances})
FIM=FIM.sort_values(by=['Feature_importance'])
FIM.tail(15)
plt.figure(figsize=(10,6))
sns.barplot(y = FIM['Features'].tail(15), x =
FIM['Feature_importance'].tail(15))
```

SAS Code

```
%macro resid_num_diag(dataset,datavar,label='requested
variable',predvar=' ',predlabel='predicted variable'); title;
data shortfourplotdataset; set &dataset;  label &datavar =
&label;  if &datavar ne .; run; proc means
data=shortfourplotdataset noprint;  var &datavar; output
out=shortfourplotoutset N=nval mean=meanval; data
shortfourplotoutset; set shortfourplotoutset;  xn=nval; CALL
SYMPUT('nval',xn);  xmean=meanval; CALL SYMPUT('meanval',xmean);
%global nvalue; %let nvalue=&nval; %global meanvalue; %let
meanvalue=&meanval; run; %if &predvar ne ' '  %then %do;
data shortfourplotdataset; set shortfourplotdataset;       label
&predvar = &predlabel;     proc sort data=shortfourplotdataset
out=shortfourplottemp;        by descending &predvar;     data
shortfourplottemp; set shortfourplottemp;
shortfourplotorder = _n_;       shortfourplotgroup = 1-
(shortfourplotorder < ceil(&nvalue/2));     proc means
data=shortfourplottemp median noprint;       by
shortfourplotgroup;       var &datavar;       output
out=shortfourplotouttemp median=medresid;     run;    data
shortfourplottempnew; merge shortfourplottemp
shortfourplotouttemp; by shortfourplotgroup;       d =
abs(&datavar-medresid);     run;   run;     proc ttest
data=shortfourplottempnew plots=none;       class
shortfourplotgroup;       var d;       ods output
TTests=shortfourplotBFtemp; title1 '(Ignore this nuisance
output)';      run;      run;     data shortfourplotBFtemp2; set
shortfourplotBFtemp;       if method = 'Pooled';       t_BF =
abs(tValue);       BF_pvalue = probt;       keep t_BF BF_pvalue;
proc print data=shortfourplotBFTemp2;       title1 'P-value for
Brown-Forsythe test of constant variance';     title2 'in '
&label ' vs. ' &predlabel;     run; %end; proc sort
data=shortfourplotdataset out=shortfourplottemp;     by
&datavar; data shortfourplottemp; set shortfourplottemp;
n=&nvalue;   expectNorm = probit((_n_-.375)/(n+.25)); proc corr
data=shortfourplottemp;   var &datavar expectNorm;   title1
'Output for correlation test of normality of ' &label;   title2
'(Check text Table B.6 for threshold)'; run; title; quit; %mend
resid_num_diag;
```

```
/* Import data */
FILENAME REFFILE '/home/u45010352/Homework/NYC-Property-Sales-
Not-Clean.csv';

PROC IMPORT DATAFILE=REFFILE replace
     DBMS=CSV
     OUT=WORK.nyc;
     GETNAMES=YES;
RUN;

/* Check hard upper bound on Sale Price */
proc univariate data=nyc;
 var SALE_PRICE;
 histogram SALE_PRICE;
run;

/*We need these data columns in the future for modeling.*/
data nyc; set nyc;
log_sale_price = log(sale_price);
log_gross_square_feet = log(gross_square_feet);
log_residential_units = log(residential_units + 1);
log_commercial_units = log(commercial_units + 1);
run;

/* Add ID variable where we sort by the sale price*/
proc sort data=nyc; by SALE_PRICE;
data nyc; set nyc; ID = _n_;

/* Plot the Sale price. */
proc sgplot data=nyc;
  scatter x=ID y=sale_price;
run;

/*There is a sustantial vaiation in the for sale price. We will
probably be using a transformation.*/

/* This is a way to create dummy variables automatically. the
"class"
   statement identifies which variables are categorical. proc
glmmod
```

```
   then creates columns for every dummy variable. The column
names
   are not informative, but when you run the data through a proc
   reg statement, each uninformative column name is paired with
   an informative label.

   In this code, I consider ALL potential explanatory variables
and place
   all categorical variables at the end. Note that we use the
outdesign
   dataset to fit our model.
*/
proc glmmod data=nyc outdesign=GLMDesign outparm=GLMParm
NOPRINT;
 class borough tax_class_at_time_of_sale sale_year sale_season ;
 model sale_price = borough residential_units commercial_units
gross_square_feet tax_class_at_time_of_sale
          crime_count sale_year sale_season building_age;
run;

/* Separate Into Training and Test Sets.
Only Fit Models to the Training Set. The variable
"Selected" separates training (0) from test (1) */
proc surveyselect data=GLMDesign seed=12345 out=nyc2
     rate=0.2 outall; /* Withold 20% for validation */
run;

data train; set nyc2;
if Selected = 0;
run;

data test; set nyc2;
if Selected = 1;
run;

/* Look at crude initial model. Notice that the variable names
are all given
   as COL1-COL41. The parameter estimates table in the output of
proc reg
   will let you know which variables correspond to which column.
*/
```

```
proc reg data=train plots(maxpoints = none)=(CooksD
RStudentByLeverage DFFITS DFBETAS);
 model sale_price = COL1-COL20 / vif collin;
store regModel;
run;

/*From the crude initial model we have a few issues we need to
fix.
 Lets start by using a log transformation on the skewed
variables.*/


proc glmmod data=nyc outdesign=GLMDesign outparm=GLMParm
NOPRINT;
 class borough tax_class_at_time_of_sale sale_year sale_season ;
 model log_sale_price = borough log_residential_units
log_commercial_units log_gross_square_feet
tax_class_at_time_of_sale
          crime_count sale_year sale_season building_age;
run;




/* Separate Into Training and Test Sets.
Only Fit Models to the Training Set. The variable
"Selected" separates training (0) from test (1) */
proc surveyselect data=GLMDesign seed=12345 out=nyc2
     rate=0.2 outall; /* Withold 20% for validation */
run;

data train; set nyc2;
if Selected = 0;
run;

data test; set nyc2;
if Selected = 1;
run;

/* Look at crude initial model with the log transformations.
Notice that the variable names are all given
```

```
   as COL1-COL41. The parameter estimates table in the output of
proc reg
   will let you know which variables correspond to which column.
*/
proc reg data=train plots(maxpoints = none)=(CooksD
RStudentByLeverage DFFITS DFBETAS);
 model log_sale_price = COL1-COL20 / vif collin;
store regModel;
run;

/*We need to adjust a couple of data points in the model.
First, we need to remove the data points that have 2017 as the
age.
These data points are missing the year built for the building.

Second, we have some strong influencital points in the data that
are also outliers.
Even with a log transformation this point still had tramendus
influence to the model.
This data point had almost 20 times more commerical units than
any other observation contained within the data. */
```

```
FILENAME REFFILE '/home/u45010352/Homework/NYC-Property-Sales-
Clean.csv';

PROC IMPORT DATAFILE=REFFILE replace
     DBMS=CSV
     OUT=WORK.nyc;
     GETNAMES=YES;
RUN;

/* Check hard upper bound on genuine */
proc univariate data=nyc;
 var SALE_PRICE;
 histogram SALE_PRICE;
```

```
run;

/*We need these data columns in the future for modeling.*/
data nyc; set nyc;
log_sale_price = log(sale_price);
log_gross_square_feet = log(gross_square_feet);
log_residential_units = log(residential_units + 1);
log_commercial_units = log(commercial_units + 1);
run;

/* Add ID variable where we sort by the sale price score */
proc sort data=nyc; by SALE_PRICE;
data nyc; set nyc; ID = _n_;


/* This is a way to create dummy variables automatically. the
"class"
   statement identifies which variables are categorical. proc
glmmod
   then creates columns for every dummy variable. The column
names
   are not informative, but when you run the data through a proc
   reg statement, each uninformative column name is paired with
   an informative label.

   In this code, I consider ALL potential explanatory variables
and place
   all categorical variables at the end. Note that we use the
outdesign
   dataset to fit our model.
*/
proc glmmod data=nyc outdesign=GLMDesign outparm=GLMParm
NOPRINT;
 class borough tax_class_at_time_of_sale sale_year sale_season ;
 model log_sale_price = borough log_residential_units
log_commercial_units log_gross_square_feet
tax_class_at_time_of_sale
          crime_count sale_year sale_season building_age;
run;
```

```
/* Separate Into Training and Test Sets.
Only Fit Models to the Training Set. The variable
"Selected" separates training (0) from test (1) */
proc surveyselect data=GLMDesign seed=12345 out=nyc2
     rate=0.2 outall; /* Withold 20% for validation */
run;

data train; set nyc2;
if Selected = 0;
run;

data test; set nyc2;
if Selected = 1;
run;

/* Look at crude initial model. Notice that the variable names
are all given
   as COL1-COL41. The parameter estimates table in the output of
proc reg
   will let you know which variables correspond to which column.
*/
proc reg data=train plots(maxpoints = none)=(CooksD
RStudentByLeverage DFFITS DFBETAS);
 model log_sale_price = COL1-COL20 / vif collin;
store regModel;
run;

proc reg data=train plots(maxpoints=none);
 model log_sale_price = COL1-COL20;
 output out= out1 r = resid p = pred;
run;

/*Our best efforts have still proven that this data will not be
best supported by a lineary regression model.
It would be best to change to a model that doesn't require such
restrictive assumptions. Before we do move on to regression
trees lets get a general estimate for using a regression model.
*/
```

```
/*We will use comparison and stepwise selection methods to find
our model*/

/*Comparison selection*/
proc reg data = train;
     model log_sale_price = COL1-COL20
                          / selection = ADJRSQ Cp AIC SBC;
     title 'Compare Selection Criteria';
run;

/*StepWise*/
proc reg data=train;
 model log_sale_price = COL1-COL20
               / selection=stepwise slentry=.05 slstay=.05;
   title1 'Stepwise Selection';
run;

/*Check Assumptions*/

proc reg data=train plots(maxpoints = none);
   model log_sale_price = col9 col4 col13 col5 col10 col14 col7
col19 col18 col20 col16 col8 col3;
   output out=out1 r=resid p=pred;
   title1 'Initial model on training data';
run;

%resid_num_diag(dataset=out1, datavar=resid,
     label='Residual', predvar=pred, predlabel='Predicted');
run;

/*Let's look at a few interactions and see if they have any
additonal effect in predicting sale price*/

data train; set train;
gross_commercial = col9 * col8;
age_crime = col13 * col20;
gross_residential = col9 * col7;
run;

proc reg data = train plots(maxplots = none);
model log_sale_price = col9 col4 col13 col5
```

```
 col10 col14 col7 col19 col18 col20 col16 col8 col3
gross_commercial age_crime gross_residential;
title1 'model with interactions';
run;

proc reg data = train plots(maxplots = none);
model log_sale_price = col9 col4 col13 col5
 col10 col14 col7 col19 col18 col16 col8 col3 gross_commercial
age_crime gross_residential;
title1 'model with interactions 2';
run;

%resid_num_diag(dataset=out1, datavar=resid,
     label='Residual', predvar=pred, predlabel='Predicted');
run;
```

/*In further analysis of the data after running our models we found some interesting points.
We had a lot of issues with commercial units influencing our model. One thought was to remove this
variable completely. However, on further thought we chose to keep it as it would cause issues with
the variable gross square feet. Instead we choose to focus the model on predicting sale price of only

```
properites with at least one residential unit. It dropped the
total observations by approxiamately 800 obsercations.
*/

/* This first line of code will need to be changed */
FILENAME REFFILE '/home/u45010352/Homework/Residential-
data.csv';



PROC IMPORT DATAFILE=REFFILE replace
     DBMS=CSV
     OUT=WORK.nyc;
     GETNAMES=YES;
RUN;

/* Check hard upper bound on genuine */
proc univariate data=nyc;
 var SALE_PRICE;
 histogram SALE_PRICE;
run;

data nyc; set nyc;
log_sale_price = log(sale_price);
log_gross_square_feet = log(gross_square_feet);
log_residential_units = log(residential_units + 1);
log_commercial_units = log(commercial_units + 1);
run;

/* Add ID variable where we sort by the sale price score */
proc sort data=nyc; by SALE_PRICE;
data nyc; set nyc; ID = _n_;

/* Plot the genunine scores. */
proc sgplot data=nyc;
  scatter x=ID y=sale_price;
run;

/* This is a way to create dummy variables automatically. the
"class"
```

```
   statement identifies which variables are categorical. proc
glmmod
   then creates columns for every dummy variable. The column
names
   are not informative, but when you run the data through a proc
   reg statement, each uninformative column name is paired with
   an informative label.

   In this code, I consider ALL potential explanatory variables
and place
   all categorical variables at the end. Note that we use the
outdesign
   dataset to fit our model.
*/
proc glmmod data=nyc outdesign=GLMDesign outparm=GLMParm
NOPRINT;
 class borough tax_class_at_time_of_sale sale_year sale_season ;
 model log_sale_price = borough log_residential_units
log_commercial_units log_gross_square_feet
tax_class_at_time_of_sale
          crime_count sale_year sale_season building_age;
run;




/* Separate Into Training and Test Sets.
Only Fit Models to the Training Set. The variable
"Selected" separates training (0) from test (1) */
proc surveyselect data=GLMDesign seed=12345 out=nyc2
     rate=0.2 outall; /* Withold 20% for validation */
run;

data train; set nyc2;
if Selected = 0;
run;

data test; set nyc2;
if Selected = 1;
run;
```

```
/* Look at crude initial model. Notice that the variable names
are all given
   as COL1-COL41. The parameter estimates table in the output of
proc reg
   will let you know which variables correspond to which column.
*/
proc reg data=train plots(maxpoints = none)=(CooksD
RStudentByLeverage DFFITS DFBETAS);
 model log_sale_price = COL1-COL20 / vif collin;
run;

proc reg data=train plots(maxpoints=none);
 model log_sale_price = COL1-COL20;
 output out= out1 r = resid p = pred;
run;



/*Comparison selection*/
proc reg data = train;
     model log_sale_price = COL1-COL20
                          / selection = ADJRSQ Cp AIC SBC;
     title 'Compare Selection Criteria';
run;

/*StepWise*/
proc reg data=train;
 model log_sale_price = COL1-COL20
                / selection=stepwise slentry=.05 slstay=.05;
   title1 'Stepwise Selection';
run;

/*Check Assumptions*/

proc reg data=train plots(maxpoints = none);
   model log_sale_price = col9 col4 col13 col5 col10 col14 col7
col19 col18 col20 col16 col8 col3;
   output out=out1 r=resid p=pred;
   title1 'Initial model on training data';
run;
```

```
%resid_num_diag(dataset=out1, datavar=resid,
     label='Residual', predvar=pred, predlabel='Predicted');
run;

/*Let's look at a few interactions and see if they have any
additonal effect in predicting sale price*/

data train; set train;
gross_commercial = col9 * col8;
age_crime = col13 * col20;
gross_residential = col9 * col7;
run;

data test; set test;
gross_commercial = col9 * col8;
gross_residential = col9 * col7;
run;

proc reg data = train plots(maxplots = none);
model log_sale_price = col9 col4 col13 col5
 col10 col14 col7 col19 col18 col20 col16 col8 col3
gross_commercial age_crime gross_residential;
title1 'model with interactions';
run;

proc reg data = train plots(maxplots = none);
model log_sale_price = col9 col4 col13 col5
 col10 col14 col7 col19 col18 col20 col16 col8 col3
gross_commercial gross_residential;
store regModel;
title1 'model with interactions 2';
run;

/*%resid_num_diag(dataset=out1, datavar=resid,
     label='Residual', predvar=pred, predlabel='Predicted');
run;*/

/*From the test results we have discoverd that two of the three
interactions did improve our model.
We will keep the interaction of gross-commerical and gross-
residential. As mentioned before our best
```

```
efforts to transform the data to meet linear assumptions has
still not worked. Our reccomendation would be
to use random-forest or a decission tree next as a means of
comparison. */

/*Let's now get the MSPR for the model.*/


data test; set test;
   log_sale_price_hat = 7.37153 +0.81521*col9
   + 0.40466*col4 - 0.0000317*col13
   + 0.12252*col5 - 0.23501*col10
   -0.08660*col14 +1.37915*col7 + 0.0452*col19
   + 0.05691*col18 - 0.03636*col16 - 0.54458*col8
   -0.14713*col3 +0.05687*gross_commercial
   - 0.17837*gross_residential;
   SqPredError = (log_sale_price - log_sale_price_hat)**2;
proc means data=test mean;
   var SqPredError;
   title1 'MSPR for test set';
run;

/* Calculate MSPR */
proc plm restore=regModel;
 score data=test out=newTest predicted;
 run;

data newTest; set newTest;
MSE = (log_sale_price - Predicted)**2;
run;

proc means data = newTest;
var MSE;
run;

proc means data= train;
var log_sale_price;
run;

data newtest2; set test;
sqPredError = (log_sale_price - 13.321)**2;
```

```
proc means data = newtest2 mean;
var sqPredError;
title1 'MSR for test set using Average'
run;

/*Now its time to use a regression tree method.*/

proc hpforest data=nyc maxtrees=100 seed=12345;
     target log_sale_price / level=interval;
     input log_residential_units log_commercial_units
crime_count building_age / level=interval;
     input borough tax_class_at_time_of_sale sale_season /
level=nominal;
     input sale_year / level=binary;
/*code file='/home/u45010352/Homework/tree2.sas';*/
run;

/* Fit a regression tree */
proc hpsplit data=nyc seed=12345 maxdepth=15 maxbranch=2;
class borough tax_class_at_time_of_sale sale_year sale_season;
model log_sale_price = borough log_residential_units
log_commercial_units
 tax_class_at_time_of_sale crime_count sale_year sale_season
building_age;
code file='/home/u45010352/Homework/tree.sas'; /* This saves the
tree to a file (need to change the path) */
run;


/* Call the test data and include the tree, this will make
predictions on the tree */

data scored;
set test;
%include '//home/u45010352/Homework/tree.sas';
run;

/* Now calculate the MSPR as we did in OLS */

data testTree;
set scored;
```

```
ASE = (log_sale_price - P_log_sale_price)**2;
run;

proc means data = testTree;
var ASE;
run;
```