NYC Properties

Kira Cozzens

Western Governors University

A. Research Question

The research question for this project is "Which property characteristics influence the sale price of NYC properties?". Answering this question allows realtors to gain a better understanding of which characteristics of properties in New York City (borough in which property is located, number of residential units at each property, number of commercial units at each property, tax class, land area of property, and the total area of the floors in the building) would affect sale prices of future properties.

The research question is based on a series of datasets from the New York City Department of Finance that covers information about properties in different NYC boroughs over a 12-month period (*Rolling Sales Data*, n.d.). Kaggle integrated these datasets into one singular set, which allows those answering the research question to analyze property characteristics across the different boroughs affect the sale price (*NYC Property Sales*, 2017).

The null hypothesis states that there is no significant relationship between the dependent variable (sale price) and the independent variables (Borough, Residential Units, Residential Units, Commercial Units, Tax_numeric, Land Square Feet, and Gross Square Feet). The null hypothesis isn't rejected if the p-values for the independent variables are greater than an alpha level of 0.05. The alternate hypothesis states that there is a significant relationship between the dependent variable (sale price) and the independent variables (Borough, Residential Units, Commercial Units, Tax_numeric, Land Square Feet, and Gross Square Feet). The null hypothesis is rejected if the p-values for the independent variables are less than or equal to an alpha level of 0.05.

B. Data Collection

The dataset contains 19 variables, not all of which are relevant to the research question. There are variables that contain further information about the sale and hyperspecific information about

the property (such as the address) that doesn't contribute to the analysis of what impacts the sale price. In this case, the relevant variables are the following:

Variable	Brief Descirption
Borough	Which NYC borough the property is located in
Residential Units	Number of residential units at each property
Commercial Units	Number of commercial units at each property
Tax Class at Present (encoded as Tax_numeric)	The tax class that each property belongs to (1-4) that describe different kinds of properties
Land Square Feet	The total square feet of the land on which the property sits
Gross Square Feet	The total area of all the floors of the building on each property
Sale Price	Price paid for the property

The data-gather methodology for this project is simple. Since the original datasets were available for the public on the New York City Department of Finance's website and on Kaggle, there was no need to actively gather data on this end. An advantage of this is that it's a simple process that requires minimum effort on the analyst's part. A disadvantage, however, is that the analyst doesn't have access to the original, raw data, which could've provided further insights.

There weren't many challenges when collecting the data. Neither site listed any problems, and the datasets were fairly clean and prepared for use. Once the dataset was found, the analysis could immediately begin.

C. Data Extraction and Preparation

-Describe data extraction and prep process with screenshots

The data is extracted from the .csv file found on the Kaggle website. The .csv file was read into Python, making the extraction process relatively short. Python is the coding environment used to clean, prepare, and analyze the data. The first step to clean the data was to check for duplicates using the *.duplicated()* function. There weren't any, but a *.drop_duplicates()* function was run just to be thorough. The advantage of this method is that it's thorough and ensures that each property is only included in the analysis once. Since this is a simple process, there don't appear to be any disadvantages.

```
#Check for duplicated data
```

```
df.duplicated()
```

```
0        False
1        False
2        False
3        False
4        False
         ...
84543    False
84544    False
84545    False
84546    False
84547    False
Length: 84548, dtype: bool
```

```
print(df.duplicated().value_counts())
```

```
False    84548
dtype: int64
```

```
df=df.drop_duplicates()
```

```
print(df.duplicated().value_counts())
```

```
False    84548
dtype: int64
```

The next step is to check for missing data using the .isnull().sum() function. This neatly sums up which variables have missing data. I also use an msno matrix to visually confirm whether or not there are any missing values. A disadvantage of this method is that it didn't account for some improperly formatted values that read as missing values (this was discovered through later

errors). These variables must then be formatted to the correct variable type and have their null values dropped. The advantage to this is that all null values are now definitely dropped.

```
1 #Check for missing data
```

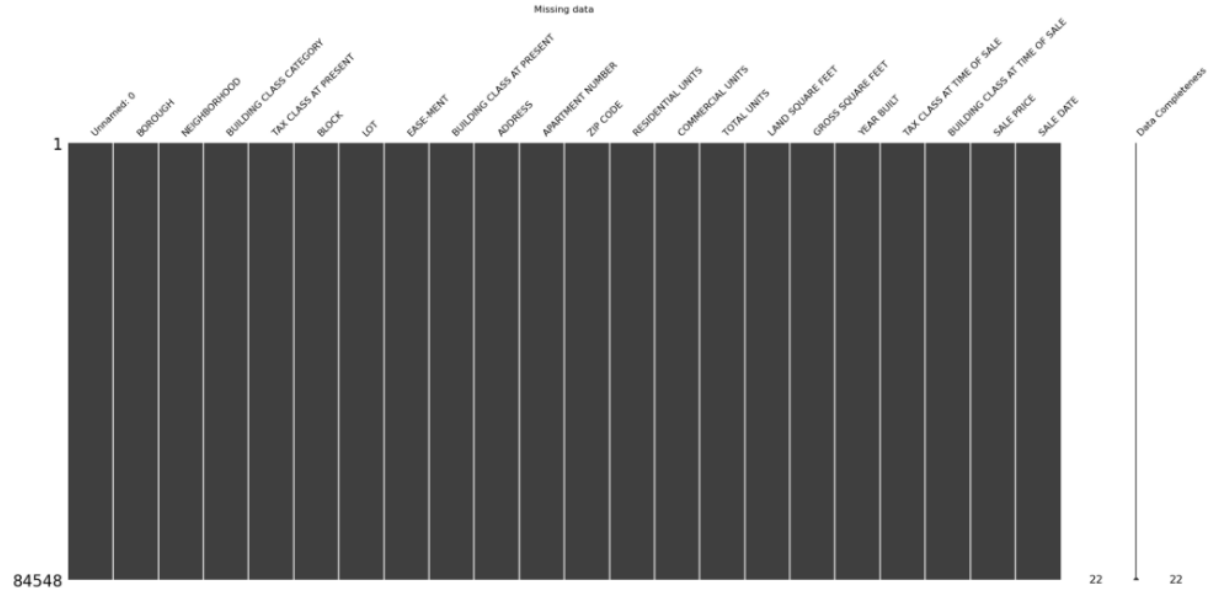
```
1 df.isnull().sum()
```

```
Unnamed: 0                        0
BOROUGH                           0
NEIGHBORHOOD                      0
BUILDING CLASS CATEGORY           0
TAX CLASS AT PRESENT              0
BLOCK                             0
LOT                               0
EASE-MENT                         0
BUILDING CLASS AT PRESENT         0
ADDRESS                           0
APARTMENT NUMBER                  0
ZIP CODE                          0
RESIDENTIAL UNITS                 0
COMMERCIAL UNITS                  0
TOTAL UNITS                       0
LAND SQUARE FEET                  0
GROSS SQUARE FEET                 0
YEAR BUILT                        0
TAX CLASS AT TIME OF SALE         0
BUILDING CLASS AT TIME OF SALE    0
SALE PRICE                        0
SALE DATE                         0
dtype: int64
```

```
import missingno as msno
```

```
import matplotlib.pyplot as plt
```

```
msno.matrix(df, fontsize=12, labels=True)
plt.title('Missing data')
plt.show()
```




```
#Convert variables to a proper format and drop hidden null values among quantitative variables
```

```
df['LAND SQUARE FEET'] = pd.to_numeric(df['LAND SQUARE FEET'], errors='coerce')
```

```
df = df.dropna(subset=['LAND SQUARE FEET'])
```

```
df['GROSS SQUARE FEET'] = pd.to_numeric(df['GROSS SQUARE FEET'], errors='coerce')
```

```
df = df.dropna(subset=['GROSS SQUARE FEET'])
```

```
df['SALE PRICE'] = pd.to_numeric(df['SALE PRICE'], errors='coerce')
```

```
df = df.dropna(subset=['SALE PRICE'])
```

Next, the applicable variables (“RESIDENTIAL UNITS”, “COMMERICAL UNITS”, “GROSS SQUARE FEET”, “LAND SQUARE FEET”) are converted to an integer format so tha the analysis can be performed. This is done by encoding the unique values to their proper numeric format. The disadvantage of this is that it’s a lengthy process. The advantage is that it definitively

formats the variables correctly. The "Tax Class at Present" variable is also encoded to a numeric value.

```
#Fix variables so that they're in integer format
```

```
df.BOROUGH.unique()
```

```
array([1, 2, 3, 4, 5], dtype=int64)
```

```
df['RESIDENTIAL UNITS'].unique()
```

```
array([   5,   10,    6,    8,   24,    3,    4,    0,    1,    2,   22,
          9,   15,   30,   35,   11,   28,    7,   18,   12,   16,   20,
         21,   19,   48,  529,  476,  317,   14,   42,  107,   31,   36,
         34,   33,   74,   29,   23,   25,  286,  256,  771,  257,   38,
        311,   41,   51,   76,   45,   72,   56,   68,   37,   50,   26,
         17,   61,   60,  894,   67,  184,   78,  181,   13,  102,  121,
         52,   27,   70,  369,   77,   40,  233,   91,   32,  109,  387,
        153,  193,   62,  146,   94,   46,   44,   84,   75,   65,   95,
         49,   63,   64,  100,   54,   43,  716,  680,   47,  179,   71,
         39,   53,   55,  283,   66,  114,   59,   89,   73,   79,   83,
        498,   81,  162,  127,  222,   99,  130,   90,   58,  159,  462,
        142,  948,  129,  889,  271,  150,  120,   57,  117,  106,   85,
        103,  118,  164,  139,  165,  122,  113,  134,  278,  135,  324,
        180,   96,  144,  152,   88,  136,  291, 1844,  198,  148,  446,
        335,  143,  128], dtype=int64)
```

```
df['RESIDENTIAL UNITS_numeric']=df['RESIDENTIAL UNITS']
```

```
dict_res={'RESIDENTIAL UNITS_numeric':{'5.':'5','10.':'10','6.':'6','8.':'8','1.':'1','3.':'3','4.':'4','0.':'0','2.':'2',
                                       '9.':'9','15.':'15','11.':'11','7.':'7','18.':'18','12.':'12','16.':'16','20.':'20',
                                      '19.':'19','14.':'14','17.':'17','13.':'13'}}
```

```
df.replace(dict_res, inplace=True)
```

```
df['COMMERCIAL UNITS'].unique()
```

```
array([   0,    1,    2,    3,   13,    5,    4,   19,   10,   14,    8,
          6,   35,   55,   17,   12,   15,    9,   23,   52,  318,   11,
        254,    7,   26,   59,   62,   42,   32,   20,   22,   28,  147,
        184,   25,  172,  436,   16, 2261,   51,   18,   21,  126],
      dtype=int64)
```

```
df['COMMERICAL UNITS_numeric']=df['COMMERCIAL UNITS']
```

```
dict_comm={'COMMERCIAL UNITS_numeric':{'0.':'0','1.':'1','2.':'2','3.':'3','13.':'13','5.':'5','4.':'4','19.':'19',
                                       '10.':'10',
                                       '14.':'14','8.':'8','6.':'6','17.':'17','12.':'12','15.':'15','9.':'9','23.':'23',
                                      '11.':'11','7.':'7','26.':'26','20.':'20','22.':'22','28.':'28','25.':'25',
                                       '16.':'16',
                                     '18.':'18','21.':'21'}}
```

```
df.replace(dict_comm, inplace=True)
```

```
df = df.dropna(subset=['GROSS SQUARE FEET'])
```

```
df['GROSS SQUARE FEET'] = df['GROSS SQUARE FEET'].replace(' - ', np.nan)  # Replace with NaN
```

```
mean_value = df['GROSS SQUARE FEET'].mean()
```

```
df['LAND SQUARE FEET'].unique()
```

```
array([  1633.,   2272.,   2369., ...,  11088., 208033.,  10796.])
```

```
df['LAND SQUARE FEET'] = df['LAND SQUARE FEET'].astype(int)
```

```
df['LAND SQUARE FEET'].unique()
```

```
array([  1633.,   2272.,   2369., ...,  11088., 208033.,  10796.])
```

```
df['LAND SQUARE FEET'] = df['LAND SQUARE FEET'].astype(int)
```

```
df['GROSS SQUARE FEET'].unique()
```

```
array([ 6440.,  6794.,  4615., ...,   977.,  2683., 64117.])
```

```
df['GROSS SQUARE FEET'] = df['GROSS SQUARE FEET'].astype(int)
```

```
#Ordinal Encoding
```

```
df['TAX CLASS AT PRESENT'].unique()
```

```
array(['2A', '2B', '2', '4', '1', '2C', '1A', '1B', '3', ' ', '1C'],
      dtype=object)
```

```
df['TAX_numeric']=df['TAX CLASS AT PRESENT']
```

```
dict_tax={'TAX_numeric':{' ':0, '1':1, '1A':2,'1B':3,'1C':4,'2':5,'2A':6,
                         '2B':7,'2C':8,'3':9,'4':10}}
```

```
df.replace(dict_tax, inplace=True)
```

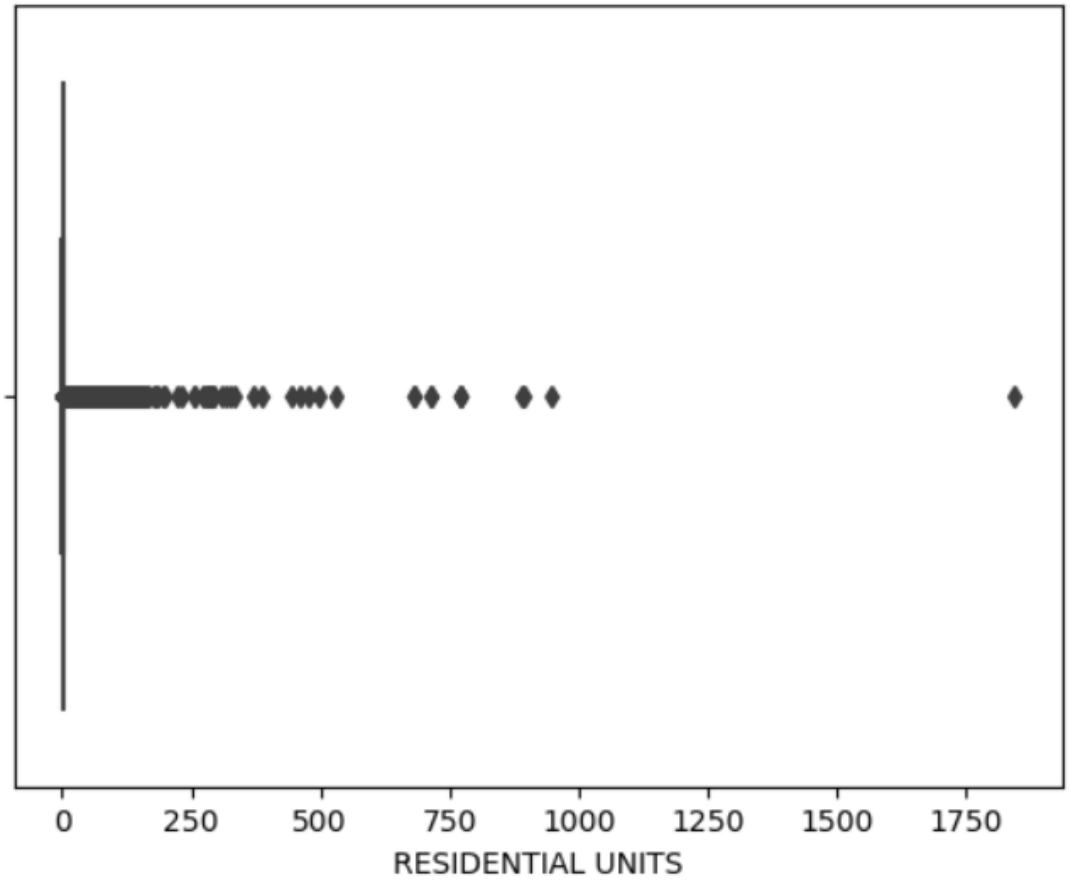
The final step is to check for outliers among all of the quantitative variables using boxplots. The advantage to this method is that it allows an analyst to visually confirm the presence of outliers. The disadvantage to this is that it requires a lot of trial and error to determine which value should be used when imputing the median.

```
]:  1 import seaborn
```

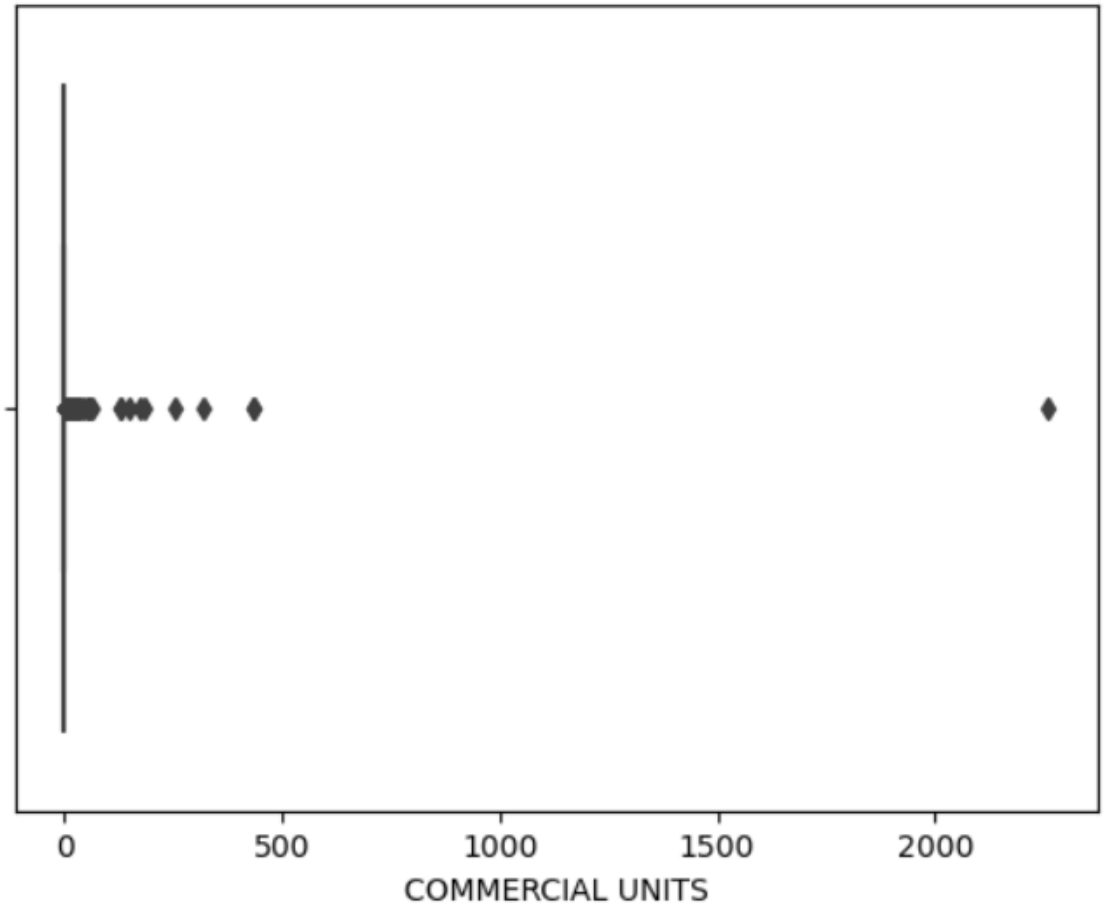
```
]:  1 from pandas import DataFrame
```

```
]:  1 import scipy.stats as stats
```

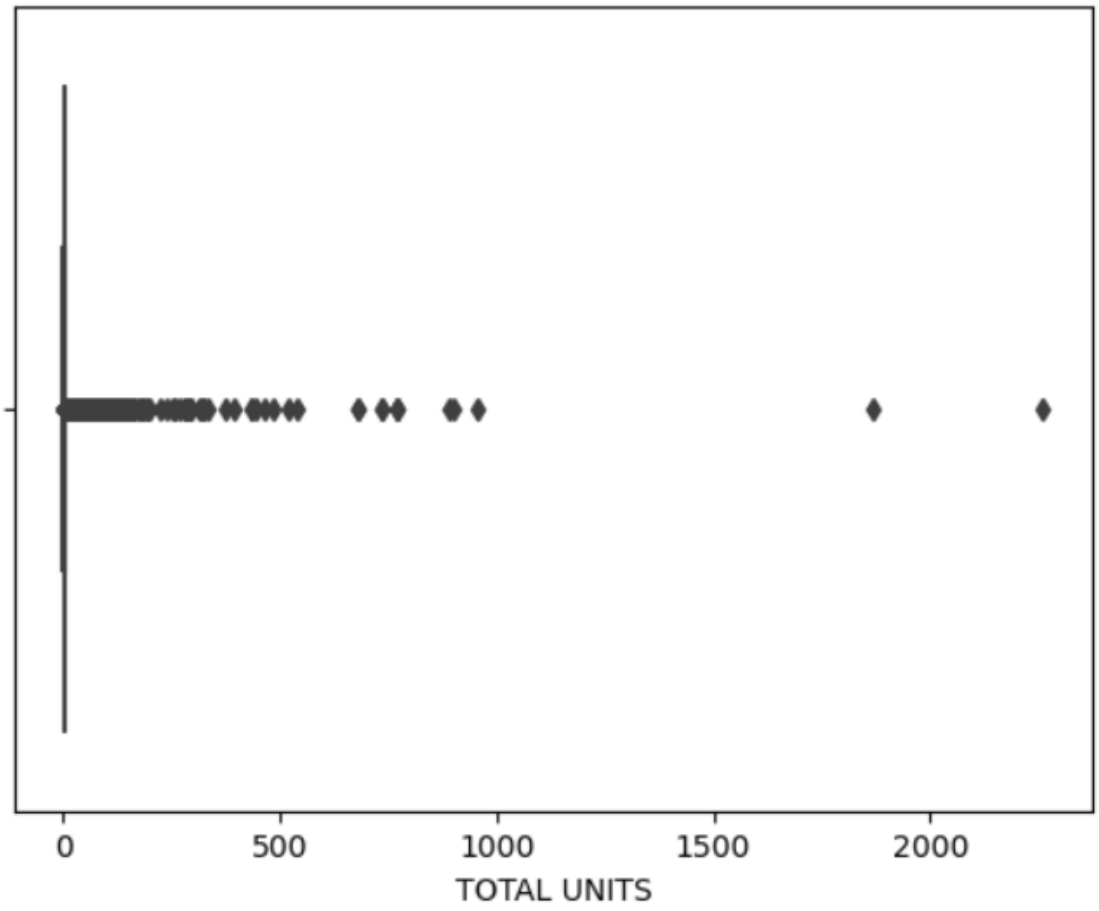
```
]:  1 boxplot1=seaborn.boxplot(x='RESIDENTIAL UNITS', data=df)
```



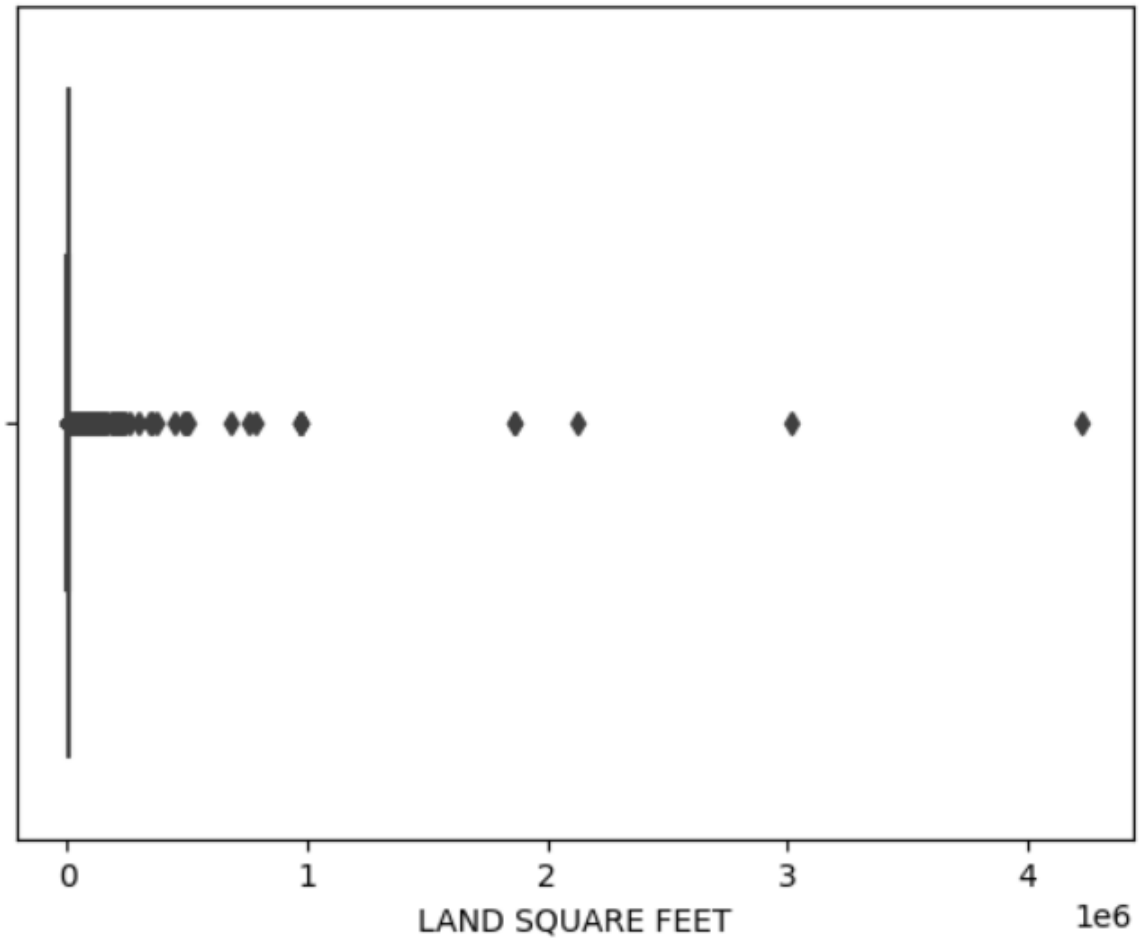
|:
```
1 boxplot2=seaborn.boxplot(x='COMMERCIAL UNITS', data=df)
```



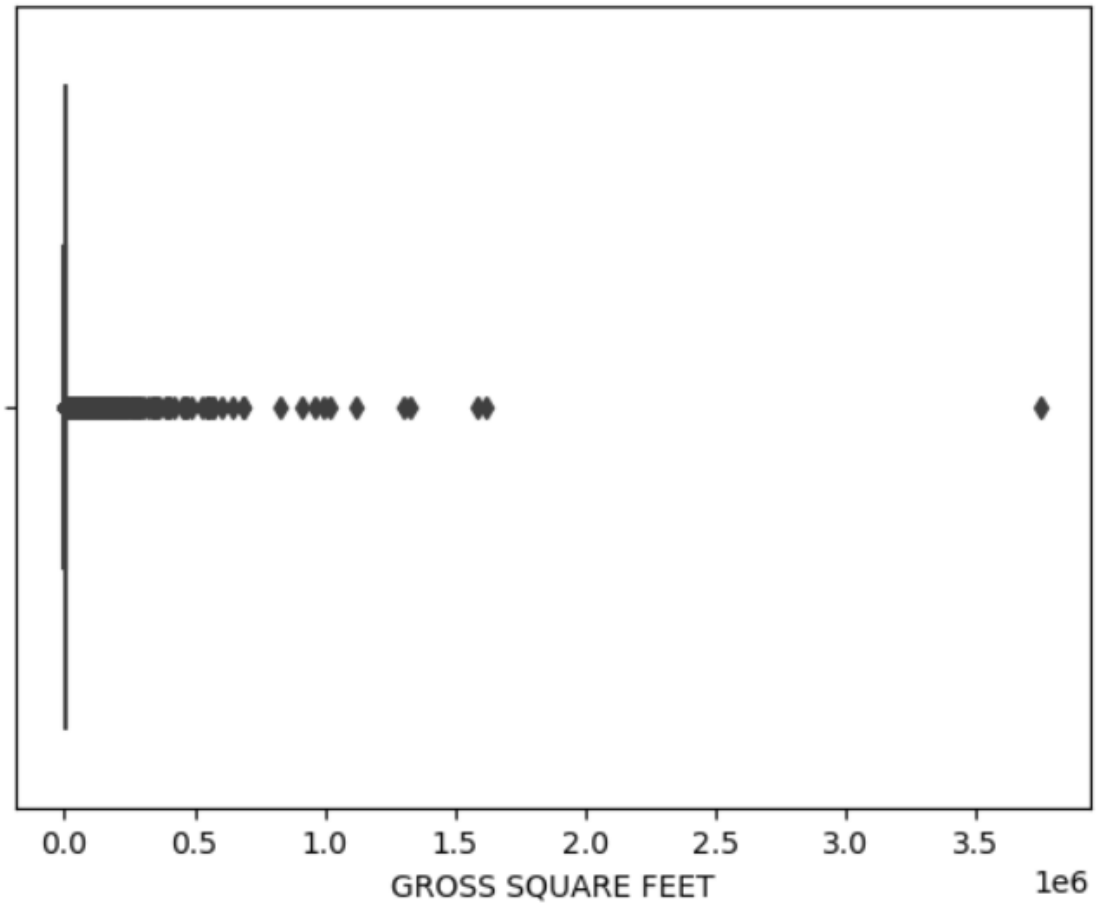
```
:  1  boxplot3=seaborn.boxplot(x='TOTAL UNITS', data=df)
```



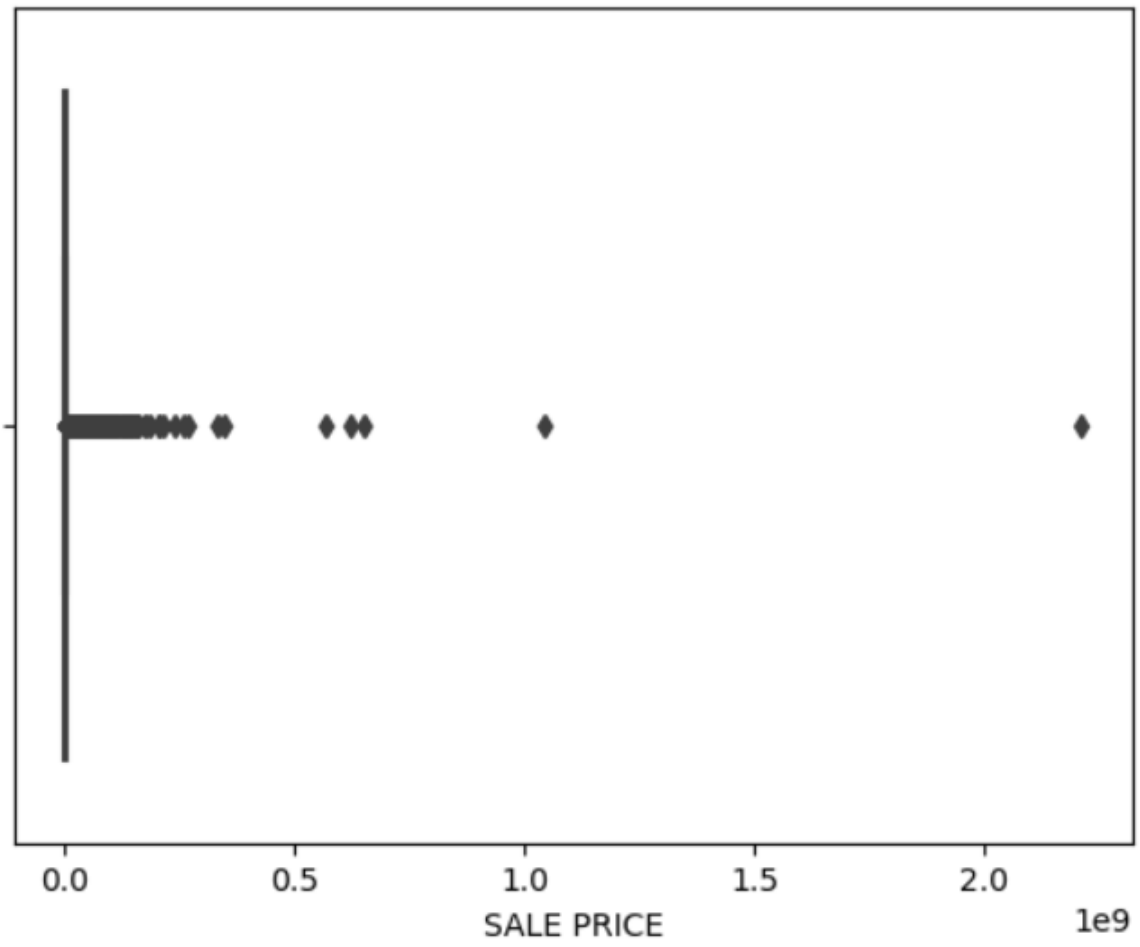
```
1 boxplot4=seaborn.boxplot(x='LAND SQUARE FEET',data=df)
```



```
1 boxplot5=seaborn.boxplot(x='GROSS SQUARE FEET',data=df)
```



```
1 boxplot6=seaborn.boxplot(x='SALE PRICE', data=df)
```

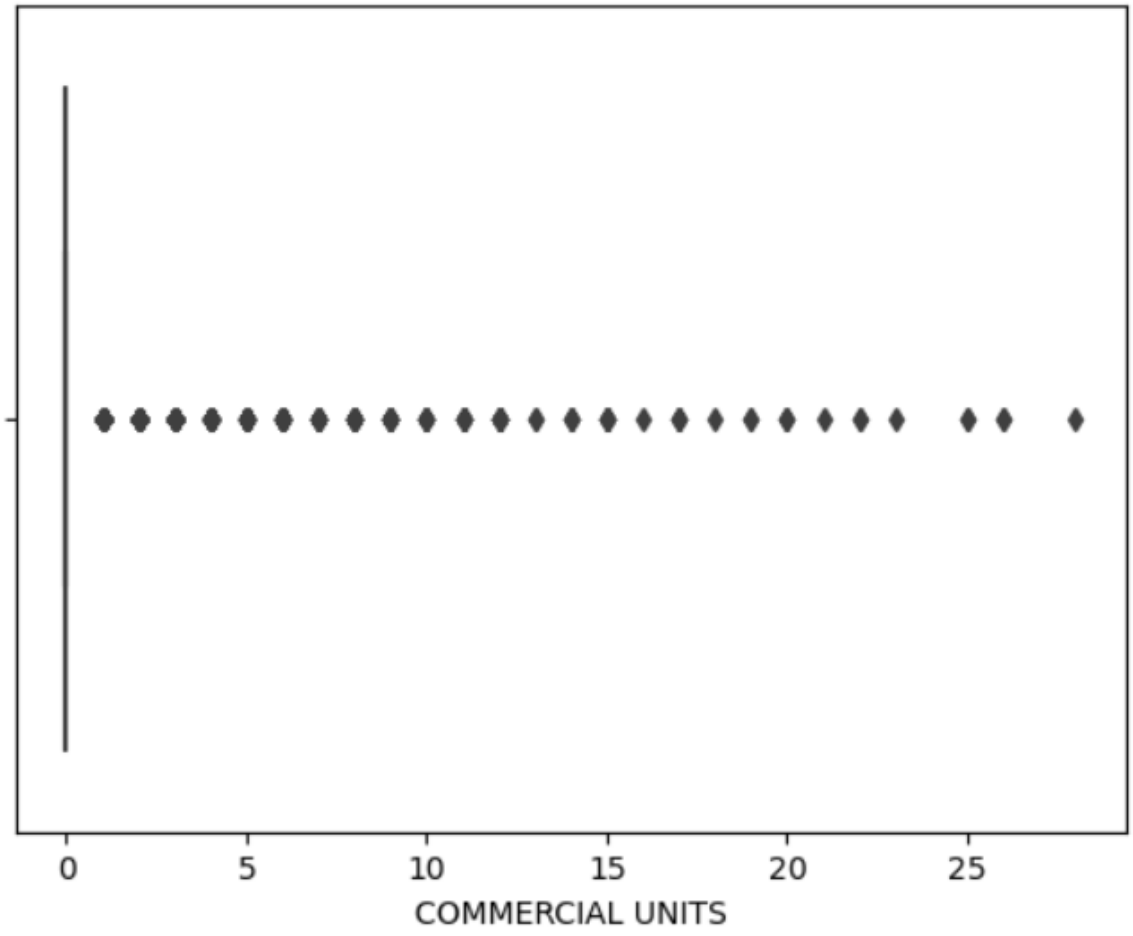


All six of these variables have large volumes of outliers, which need to be resolved. This is done by imputing the median value of each variable so that there is a reasonable volume of outliers. The advantage of this method is that the data won't be skewed, which is an effect of too many outliers. A disadvantage, as previously mentioned, is that the boxplots make it difficult to determine what the minimum and/or maximum value should be for the imputation, which means that a lot of trial and error is involved.

```
median2=np.median(df['COMMERCIAL UNITS'])
```

```
df['COMMERCIAL UNITS']=np.where(df['COMMERCIAL UNITS']>30,
                                median2, df['COMMERCIAL UNITS'])
```

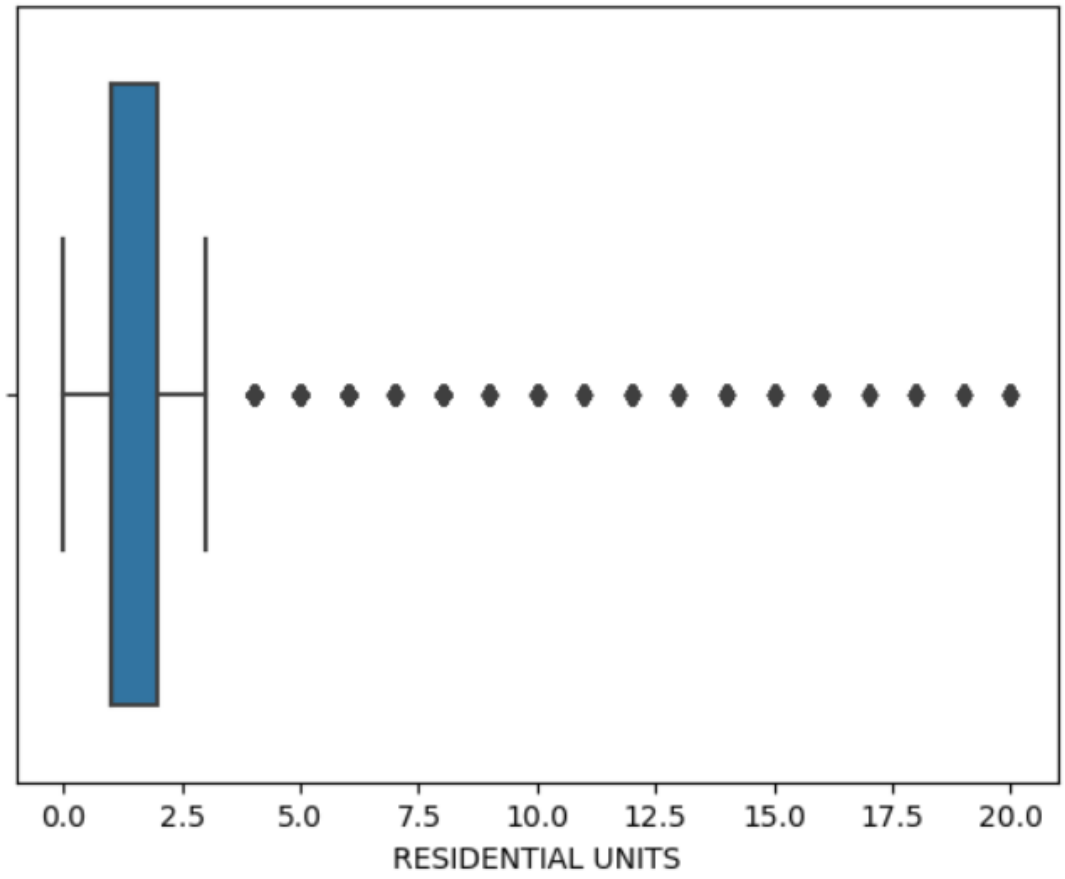
```
boxplot_n2=seaborn.boxplot(x='COMMERCIAL UNITS', data=df)
```



```
median=np.median(df['RESIDENTIAL UNITS'])
```

```
df['RESIDENTIAL UNITS']=np.where(df['RESIDENTIAL UNITS']>20,
                         median, df['RESIDENTIAL UNITS'])
```

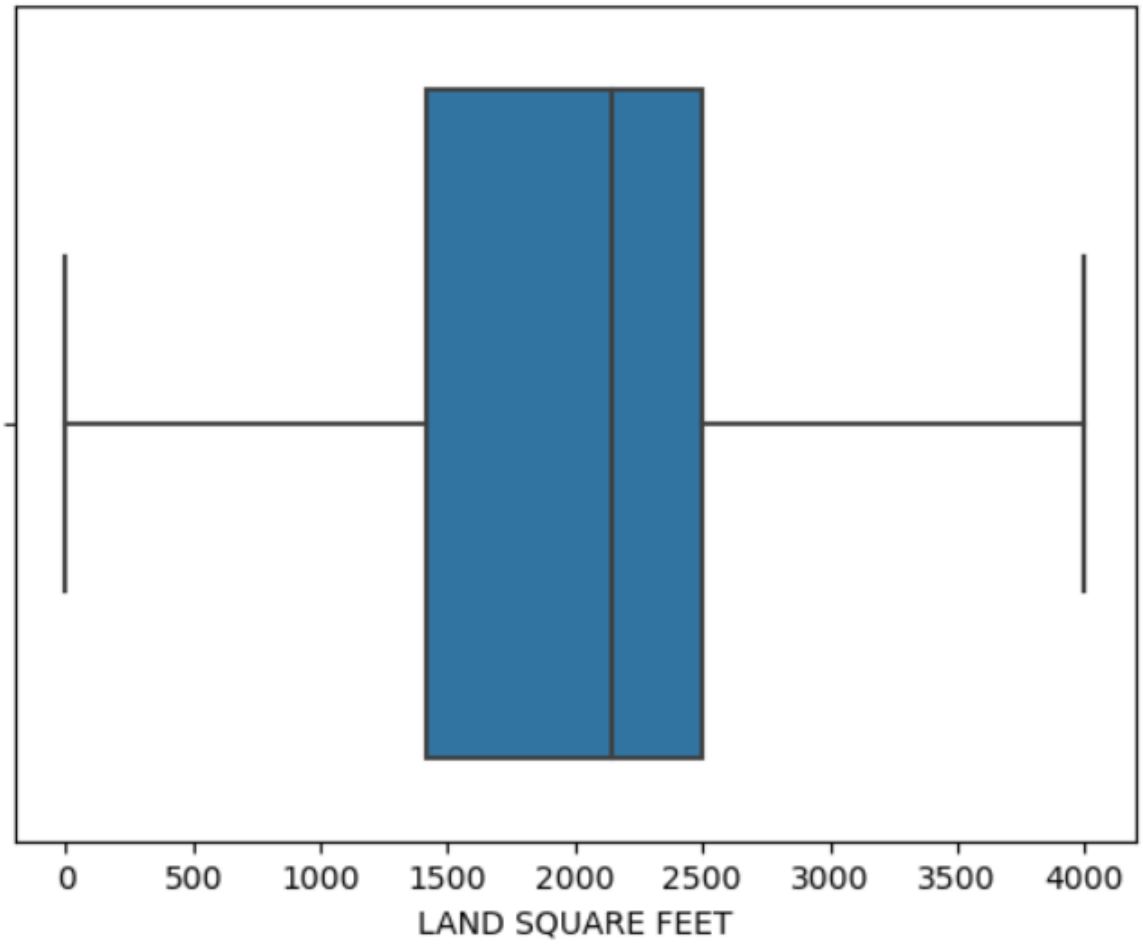
```
boxplot_n1=seaborn.boxplot(x='RESIDENTIAL UNITS',data=df)
```



```
median4=np.median(df['LAND SQUARE FEET'])
```

```
df['LAND SQUARE FEET']=np.where(df['LAND SQUARE FEET']>4000,
                                median4, df['LAND SQUARE FEET'])
```

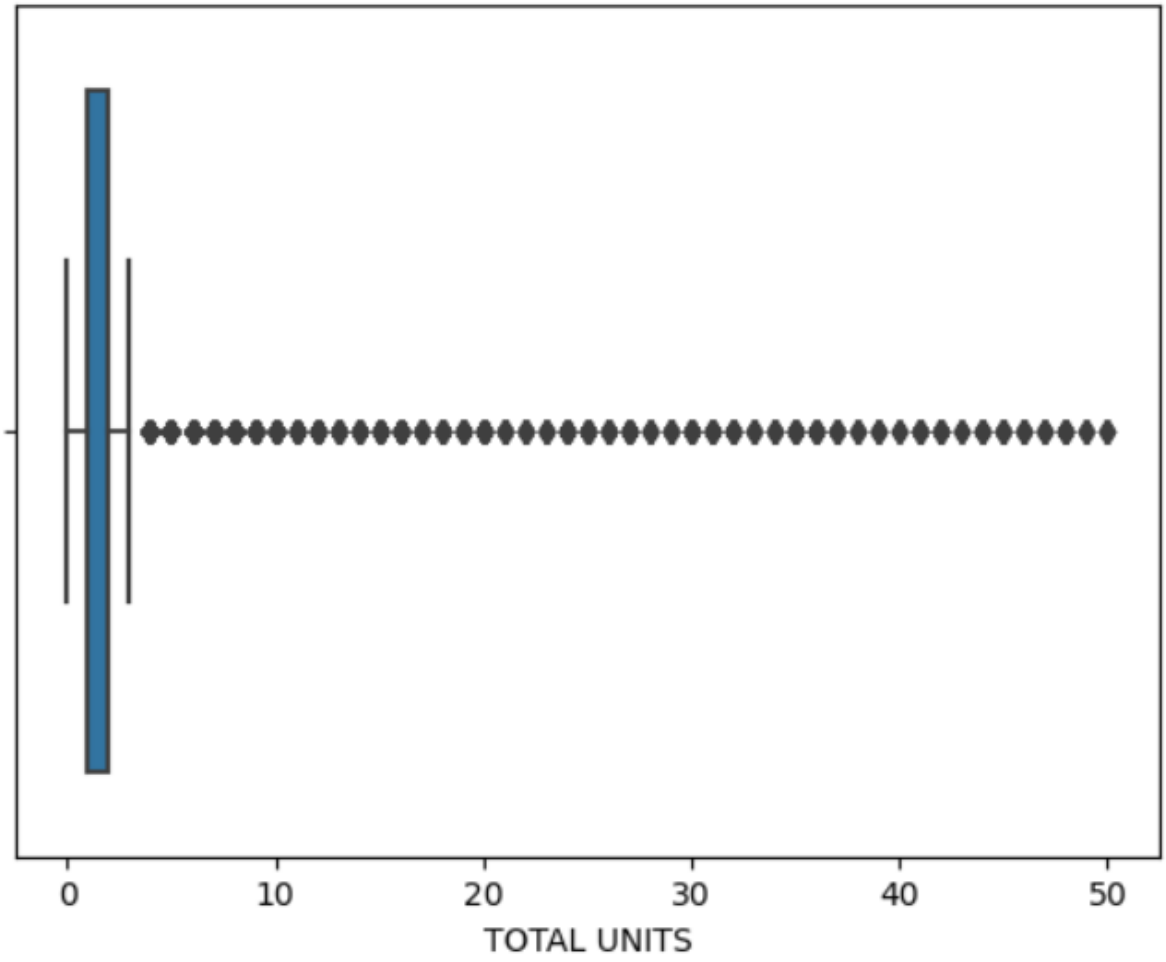
```
boxplot_n4=seaborn.boxplot(x='LAND SQUARE FEET', data=df)
```



```
median3=np.median(df['TOTAL UNITS'])
```

```
df['TOTAL UNITS']=np.where(df['TOTAL UNITS']>50,
                           median3, df['TOTAL UNITS'])
```

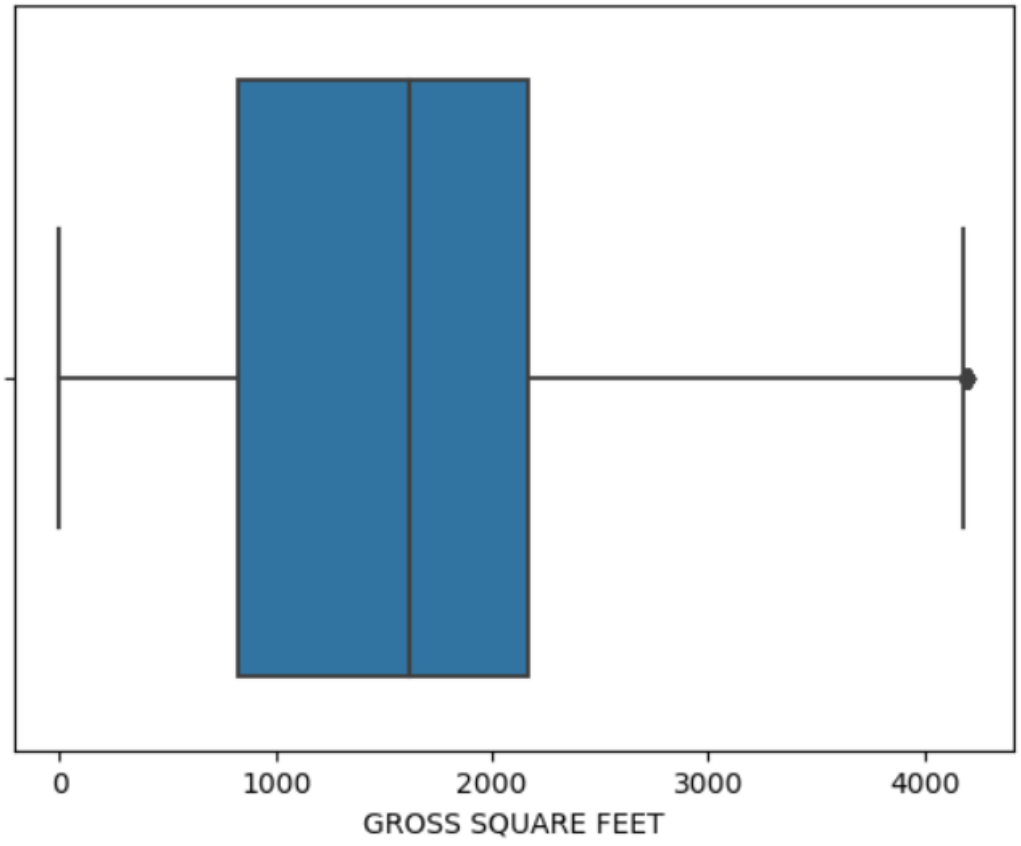
```
boxplot_n3=seaborn.boxplot(x='TOTAL UNITS', data=df)
```



```
median5=np.median(df['GROSS SQUARE FEET'])
```

```
df['GROSS SQUARE FEET']=np.where(df['GROSS SQUARE FEET']>4200,
                                 median5, df['GROSS SQUARE FEET'])
```

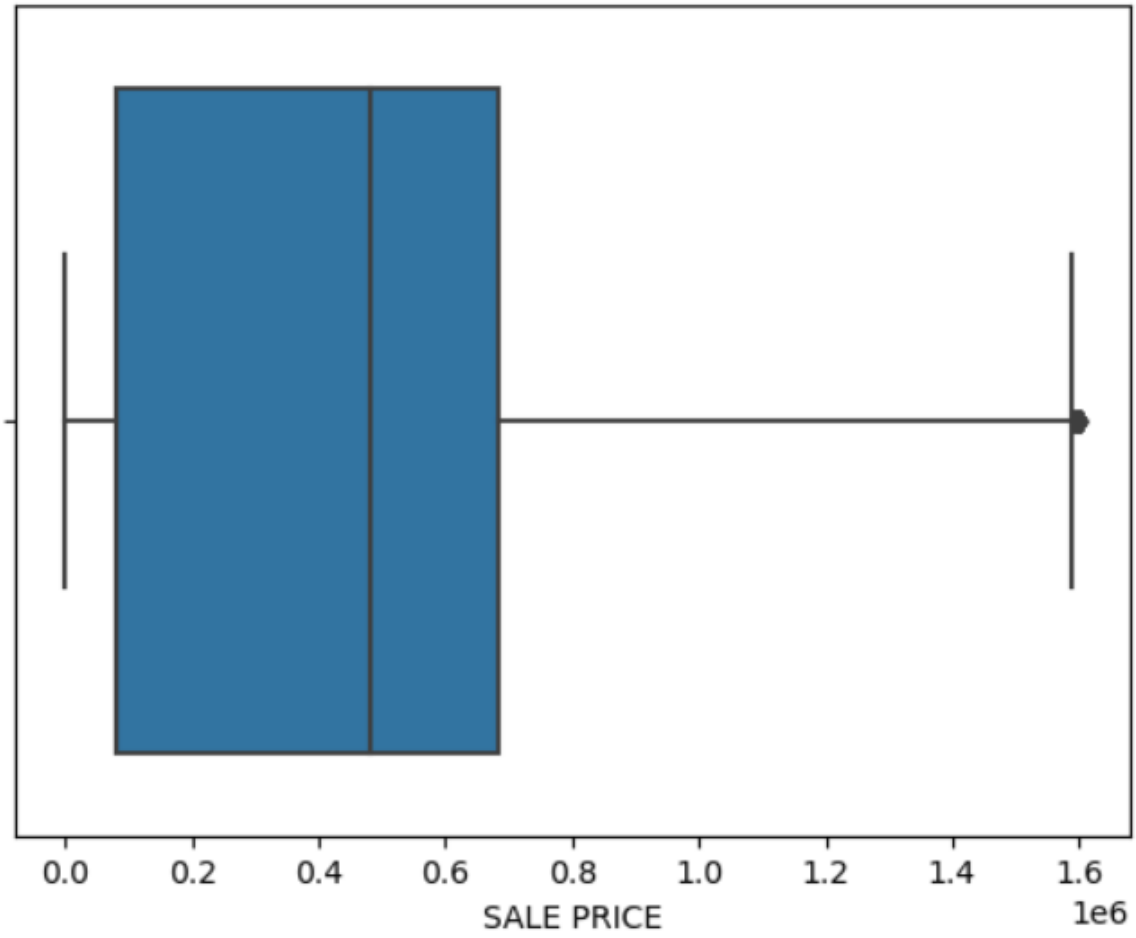
```
boxplot_n5=seaborn.boxplot(x='GROSS SQUARE FEET', data=df)
```



```
median6=np.median(df['SALE PRICE'])
```

```
df['SALE PRICE']=np.where(df['SALE PRICE']>1600000,
                          median6, df['SALE PRICE'])
```

```
boxplot_n6=seaborn.boxplot(x='SALE PRICE', data=df)
```



D. Analysis

The analysis technique used to answer the research question is multiple regression. This is done because multiple regression analyzes the impact of quantitative and qualitative independent variables on a quantitative dependent variable. This involves setting X as the independent variable that contains all of the property characteristics, “BOROUGH” ,“RESIDENTIAL

UNITS", "COMMERCIAL UNITS", "LAND SQUARE FEET","GROSS SQUARE FEET", and "TAX_numeric" and setting y as the dependent variable "SALE PRICE". These are then placed into a multiple regression model, where the p-values are used to determine which variables have a significant influence on sale price. An advantage to this technique is that it's simple to implement. A disadvantage is that it provides a limited glimpse into the data, as it only analyzes which independent variables have a significant influence on the dependent variable.

```
[ ]: 1 import statsmodels.formula.api as smf

[ ]: 1 from sklearn import linear_model

[ ]: 1 import statsmodels.api as sm

[ ]: 1 X=df[['BOROUGH','RESIDENTIAL UNITS','COMMERCIAL UNITS',
     2       'LAND SQUARE FEET','GROSS SQUARE FEET','TAX_numeric']]

[ ]: 1 y=df['SALE PRICE']

[ ]: 1 model=smf.ols('y~X', data=df)

[ ]: 1 res=model.fit()

[ ]: 1 print(res.summary())
```

```
                            OLS Regression Results
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.052
Model:                            OLS   Adj. R-squared:                  0.052
Method:                 Least Squares   F-statistic:                     441.0
Date:                Sat, 15 Jul 2023   Prob (F-statistic):               0.00
Time:                        23:50:41   Log-Likelihood:            -6.8728e+05
No. Observations:               48244   AIC:                         1.375e+06
Df Residuals:                   48237   BIC:                         1.375e+06
Df Model:                           6
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept   1.811e+05   8371.894     21.638      0.000    1.65e+05    1.98e+05
X[0]        8.795e+04   2054.530     42.806      0.000    8.39e+04     9.2e+04
X[1]        7059.6424   1056.201      6.684      0.000    4989.474    9129.810
X[2]        2530.8511   2364.956      1.070      0.285   -2104.494    7166.196
X[3]         -25.0549      2.065    -12.131      0.000     -29.103     -21.007
X[4]          32.9606      2.196     15.011      0.000      28.657      37.264
X[5]       -6450.9245    749.658     -8.605      0.000   -7920.265   -4981.584
==============================================================================
Omnibus:                     3194.669   Durbin-Watson:                   1.421
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3871.056
Skew:                           0.691   Prob(JB):                         0.00
Kurtosis:                       3.125   Cond. No.                     1.40e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.4e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

From there, the mean squared error (MSE) is calculated. This value helps to determine the accuracy of the multiple regression model by determining how close the predicted values were to the actual values. Since the MSE is large (138,470,011,655.5684) , it's assumed that the percentage of error is large. The advantage of this is that it's an easy calculation to perform, but a disadvantage is that it's only one way to analyze the accuracy of the model.

```
1 from sklearn.linear_model import LinearRegression
```

```
1 from sklearn.metrics import mean_squared_error
```

```
1 lin_regress=LinearRegression()
```

```
1 lin_regress.fit(X,y)
```

```
LinearRegression()
```

```
1 y_pred=lin_regress.predict(X)
```

```
1 y_pred
```

```
array([278167.82104213, 291005.00862662, 266787.03549798, ...,
       577637.87564476, 573865.51527501, 584389.70537782])
```

```
1 mse=mean_squared_error(y,y_pred)
```

```
1 X_=df[['BOROUGH','RESIDENTIAL UNITS','COMMERCIAL UNITS',
2        'TAX_numeric','LAND SQUARE FEET','GROSS SQUARE FEET']]
```

```
1 print(mse)
```

```
138470011655.5684
```

Another way to analyze the accuracy of the model is the R^2 value, found in the model summary above. The R^2 is 0.052, which means that the model only explains 5.2% of the variability throughout the variables. While this value could be better, it isn't terrible as it does explain at least some of the variability. Similar to the previous techniques, an advantage is that this value is easy to calculate, since it's included in the model summary, and a disadvantage is that it's not the only way to determine a model's value (Rowe, n.d.).

E. Data Summary and Implications

The null hypothesis for the multiple regression states, “The null hypothesis states that there is no significant relationship between the dependent variable (sale price) and the independent variables (Borough, Residential Units, Residential Units, Commercial Units, Tax_numeric, Land Square Feet, and Gross Square Feet). The null hypothesis isn’t rejected if the p-values for the independent variables are greater than an alpha level of 0.05.” Thus, in order to reject the null hypothesis, (signifying that the independent variables have a significant influence on the dependent variable), the p-values need to be equal to or less than 0.05. All of the variables, except for “COMMERICAL UNITS”, have a p-value of 0.00. “COMMERCIAL UNITS” has a p-value of 0.285, which means that the variables with a significant influence on sale price are “BOROUGH”, “RESIDENTIAL UNITS”, “LAND SQUARE FEET”,”GROSS SQUARE FEET”, and “TAX_numeric”. A limitation of this analysis is that since it has a high MSE value, it doesn’t appear to be a very accurate way of answering the research question.

An action that a future analyst could perform would be to remove “COMMERICAL UNITS” from the model, and analyze the MSE value and R^2 values to see if they improve. Another direction of study could be to once again run a multiple regression analysis, but instead focus on the location of the properties, using variables such as “NEIGHBORHOOD”, “ZIP CODE”, and “BUILDING CLASS AT TIME OF SALE”. One could also perform a Principle Component Analysis (PCA) to determine how to reduce the dimensions of the dataset.

F. Sources

See References.

References

NYC Property Sales. (2017, September 22). Kaggle. https://www.kaggle.com/datasets/new-york-city/nyc-property-sales

Rolling sales data. (n.d.). https://www.nyc.gov/site/finance/taxes/property-rolling-sales-data.page

Rowe, W. (n.d.). *Mean square error & R2 score clearly explained*. BMC Blogs. https://www.bmc.com/blogs/mean-squared-error-r2-and-variance-in-regression-analysis/#:~:text=There%20is%20no%20correct%20value,means%20the%20model%20is%20perfect.