Final Data Project

Marissa Baietto and Aadhi Subbiah

The Dataset

- Source: Kaggle (link: www.kaggle.com/datasets/samdeeplearning/vt-nh-real-estate)
- Originally obtained from MLS.com
- 3 towns in Vermont
- Data last updated in 2017
- Originally had 137 houses before cleaned.
- Numeric and categorical variables
  - # of bathrooms
  - Garage type

Variables

```
df['house_age'] = 2017 - df['year_built']
```

bedrooms_total
baths_total
acres
sq_foot_tot_fn
tax_gross_amount
assessment_value_town
garage_capacity
address
city

garage_type
house_age
total_stories
surveyed
water_body_type
water_frontage_length
rooms_total
garage
flood_zone

easements
current_use
covenants
basement_access_type
basement
price_closed

Describing Our Project

- Expectations and Patterns of Housing Data
  - Predict house price
  - Best factors: square footage, number of bedrooms and bathrooms, and location relative to water.
- Potential Implications
  - Optimize housing market predictions.
  - Explain trends in house prices
- Benefits of this Project
  - Buying or selling a house
  - Know which factors to look at
  - Housing inflation

Removing Duplicates

There are no duplicate rows. The only column that should have no duplicates is 'address'. There appeared to be a duplicate in this column, but the addresses were different when we looked at the rows in question.

```
duplicateRows = df[df.duplicated()]
print(duplicateRows)
```

```
Empty DataFrame
```

```
duplicateRows = df[df.duplicated('address')]
print(duplicateRows)
```

```
    id  bedrooms_total  baths_total  acres  sq_ft_tot_fn  \
21  22               3            3   15.4          2926
97  98               3            2    4.3          1764

    assessment_value_town  garage_capacity        address
21               591330.0              2.0  529 Stage Road
97               312450.0              2.0  19 Garnet Hill
```

Our process for checking NaN

1. Use the df['column_name'].unique() to see if there are any NaN values
2. Used df[df['column_name'].isnull()] to see which rows had the NaN values
3. Cleaned the data
   1. If there were numbers we could not estimate, we removed these rows from the dataset
   2. If the column had object datatypes, we replaced NaN with 'Unknown'
   3. If the column had a string values we could estimate, we replaced NaN with a more suitable value
   4. If the entire column was NaN, we removed the column from the dataset.

```
df['tax_gross_amount'].unique()
```

```
df[df['tax_gross_amount'].isnull()]
```

```
df[df['assessment_value_town'].isnull()]
```

	id	bedrooms_total	baths_total	acres	sq_ft_
7	8	3	3	10.33	

```
df = df.drop([7])
```

```
df['garage_type'] =
    df['garage_type'].fillna('Unknown')
```

```
df['water_body_type'] = df['water_body_type'].fillna('None')
```

```
df = df.drop(columns=['common_land_acres'])
```

List of modifications we made

- Filled in ‘acres’ that’s NaN with 0.
- **Took out 6 rows that had NaN for ‘tax_gross_amount’**
- **Took out 8 rows that had NaN for ‘assessment_value_town’**
- Said ‘Unknown’ for garage type NaN
- Said ‘None’ for water body type NaN
- Filled in 0 for water frontage length NaN
- Filled in ‘No’ for easements NaN
- Filled in ‘Unknown’ for current use NaN
- **Removed ‘common_land_acres’, ‘season’, and ‘short_sale’ columns entirely because every entry was either NaN or ‘no’**
- Filled ‘none’ for basement access type NaN
- **Removed a row where ‘basement’ was NaN**

Checking datatypes

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 122 entries, 0 to 136
Data columns (total 27 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   id                     122 non-null    int64
 1   bedrooms_total         122 non-null    int64
 2   baths_total            122 non-null    int64
 3   acres                  122 non-null    float64
 4   sq_ft_tot_fn           122 non-null    int64
 5   tax_gross_amount       122 non-null    float64
 6   assessment_value_town  122 non-null    float64
 7   garage_capacity        122 non-null    float64
 8   address                122 non-null    object
 9   city                   122 non-null    object
 10  garage_type            122 non-null    object
 11  year_built             122 non-null    int64
 12  total_stories          122 non-null    float64
 13  surveyed               122 non-null    object
 14  seasonal               122 non-null    object
 15  water_body_type        122 non-null    object
 16  water_frontage_length  122 non-null    int64
 17  short_sale             122 non-null    object
 18  rooms_total            122 non-null    int64
 19  garage                 122 non-null    object
 20  flood_zone             122 non-null    object
 21  easements              122 non-null    object
 22  current_use            122 non-null    object
 23  covenants              122 non-null    object
 24  basement_access_type   122 non-null    object
 25  basement               122 non-null    object
 26  price_closed           122 non-null    int64
dtypes: float64(5), int64(8), object(14)
```

```
df.iloc[[93,16]] = 1228
df.iloc[[106,16]] = 1228
df.iloc[[119,16]] = 1047
```

```
df['water_frontage_length'] = pd.to_numeric(df['water_frontage_length'])
```

There was one variable, ‘water_frontage_length’, that was an object when it should have been numeric because three rows had a comma - such as “1,228”. We replaced the three values with the integer value.

Correlations of Numeric Values

	bedrooms_total	baths_total	acres	sq_ft_tot_fn	tax_gross_amount	assessment_value_town	garage_capacity	house_age	total_stories	water_frontage_length	rooms_total	price_closed
bedrooms_total	1.000	0.633	0.284	0.641	0.537	0.457	0.195	0.368	0.202	0.223	0.639	0.466
baths_total	0.633	1.000	0.330	0.774	0.572	0.474	0.300	0.087	0.277	0.232	0.571	0.500
acres	0.284	0.330	1.000	0.299	0.518	0.385	0.165	0.044	0.086	-0.019	0.205	0.478
sq_ft_tot_fn	0.641	0.774	0.299	1.000	0.710	0.642	0.366	0.168	0.317	0.249	0.724	0.567
tax_gross_amount	0.537	0.572	0.518	0.710	1.000	0.868	0.327	0.162	0.295	0.147	0.539	0.840
assessment_value_town	0.457	0.474	0.385	0.642	0.868	1.000	0.366	0.185	0.341	0.091	0.479	0.758
garage_capacity	0.195	0.300	0.165	0.366	0.327	0.366	1.000	-0.037	0.065	-0.002	0.327	0.359
house_age	0.368	0.087	0.044	0.168	0.162	0.185	-0.037	1.000	0.003	0.119	0.234	0.150
total_stories	0.202	0.277	0.086	0.317	0.295	0.341	0.065	0.003	1.000	0.009	0.265	0.273
water_frontage_length	0.223	0.232	-0.019	0.249	0.147	0.091	-0.002	0.119	0.009	1.000	0.312	0.293
rooms_total	0.639	0.571	0.205	0.724	0.539	0.479	0.327	0.234	0.265	0.312	1.000	0.506
price_closed	0.466	0.500	0.478	0.567	0.840	0.758	0.359	0.150	0.273	0.293	0.506	1.000

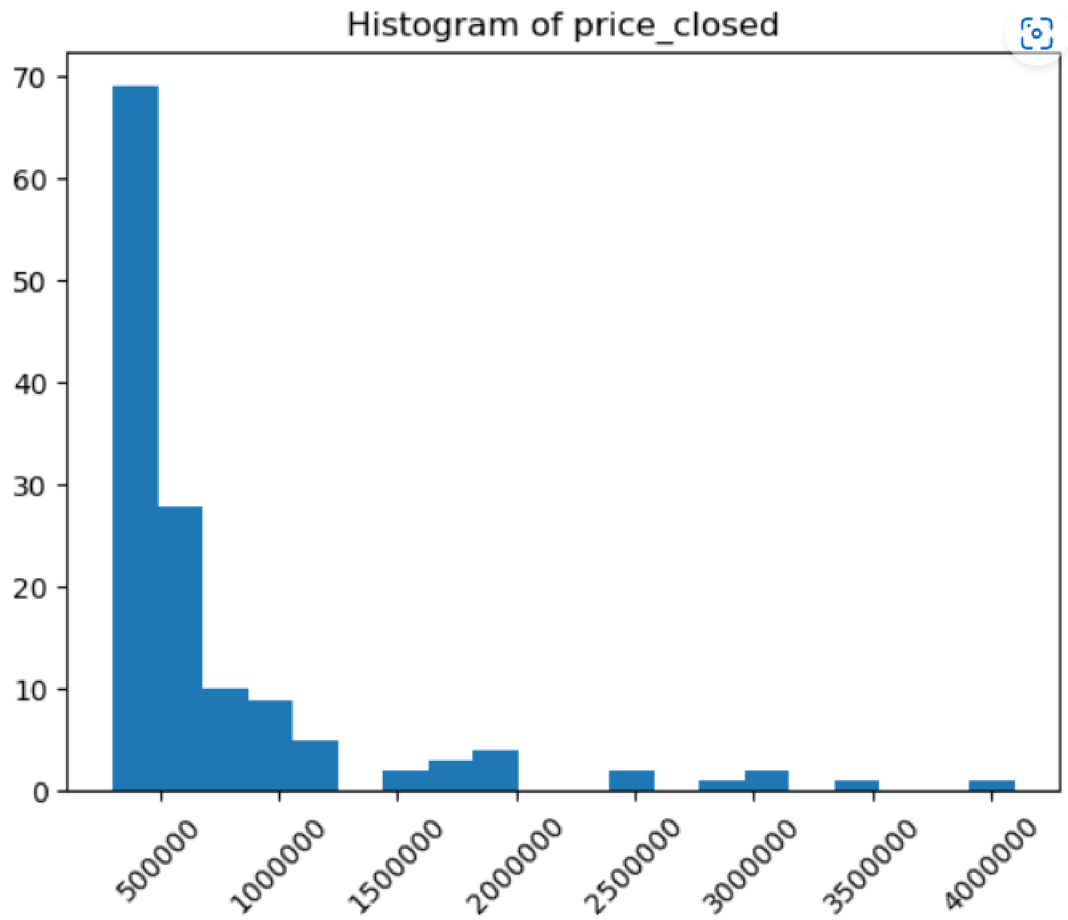
Strong correlations: sq_ft_tot_fn, tax_gross_amount,assessment_value_town,rooms_total
Moderate correlations: bedrooms_total,acres,baths_total,total_stories
Weak correlations: garage_capacity,house_age,water_frontage_length

Correlations of Categorical Values

	garage_type	surveyed	water_body_type	garage	flood_zone	easements	current_use	basement_access_type	basement	covenants	price_closed
garage_type	1.000	0.226	-0.015	0.803	-0.046	-0.043	0.043	-0.073	0.133	0.136	0.147
surveyed	0.226	1.000	0.149	0.169	-0.012	-0.050	0.067	-0.076	-0.128	0.222	0.167
water_body_type	-0.015	0.149	1.000	-0.062	0.389	0.198	0.089	-0.071	0.087	-0.003	0.086
garage	0.803	0.169	-0.062	1.000	-0.092	-0.062	0.086	-0.053	0.147	-0.005	0.181
flood_zone	-0.046	-0.012	0.389	-0.092	1.000	0.180	-0.099	0.044	0.087	-0.074	-0.107
easements	-0.043	-0.050	0.198	-0.062	0.180	1.000	-0.079	-0.128	0.053	0.210	-0.098
current_use	0.043	0.067	0.089	0.086	-0.099	-0.079	1.000	0.107	0.040	-0.156	0.441
basement_access_type	-0.073	-0.076	-0.071	-0.053	0.044	-0.128	0.107	1.000	0.284	-0.108	-0.001
basement	0.133	-0.128	0.087	0.147	0.087	0.053	0.040	0.284	1.000	-0.003	-0.001
covenants	0.136	0.222	-0.003	-0.005	-0.074	0.210	-0.156	-0.108	-0.003	1.000	0.049
price_closed	0.147	0.167	0.086	0.181	-0.107	-0.098	0.441	-0.001	-0.001	0.049	1.000

Strong correlations: None
Moderate correlations: current_use
Weak correlations: garage_type, surveyed, water_body_type, garage, flood_zone, easements, basement_access_type, basement, covenants

Price Closed

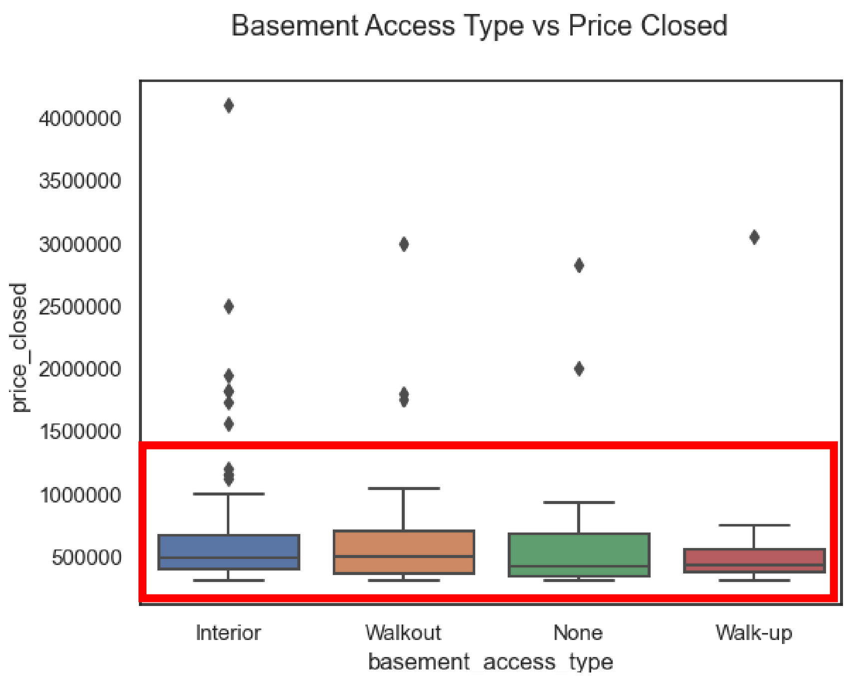



```
df["price_closed"].describe().apply(lambda x: format(x, 'f'))
```

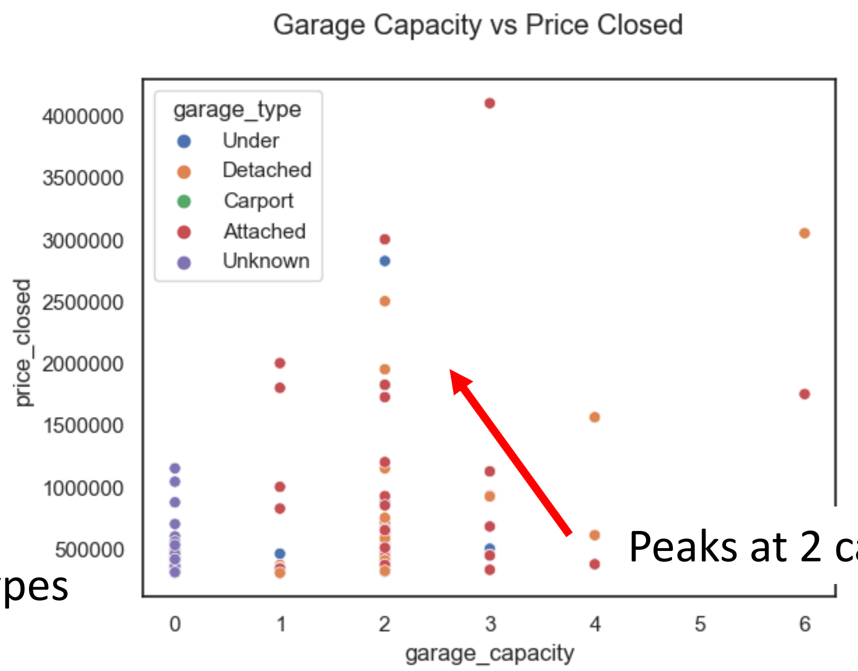
```
count        137.000000
mean      736181.919708
std       667017.709610
min       300000.000000
25%       372500.000000
50%       485000.000000
75%       750000.000000
max      4100000.000000
```

The median, $485,000, is almost $100,000 more than the median house price of the average United States home, which is ~$335,000 according to Zillow. Either these three towns are the most expensive towns in Vermont, or Vermont is a more expensive state to live in. The top 25% prices lie between $750,000 and $4,100,000, which indicates that most of the outliers are in the high prices.

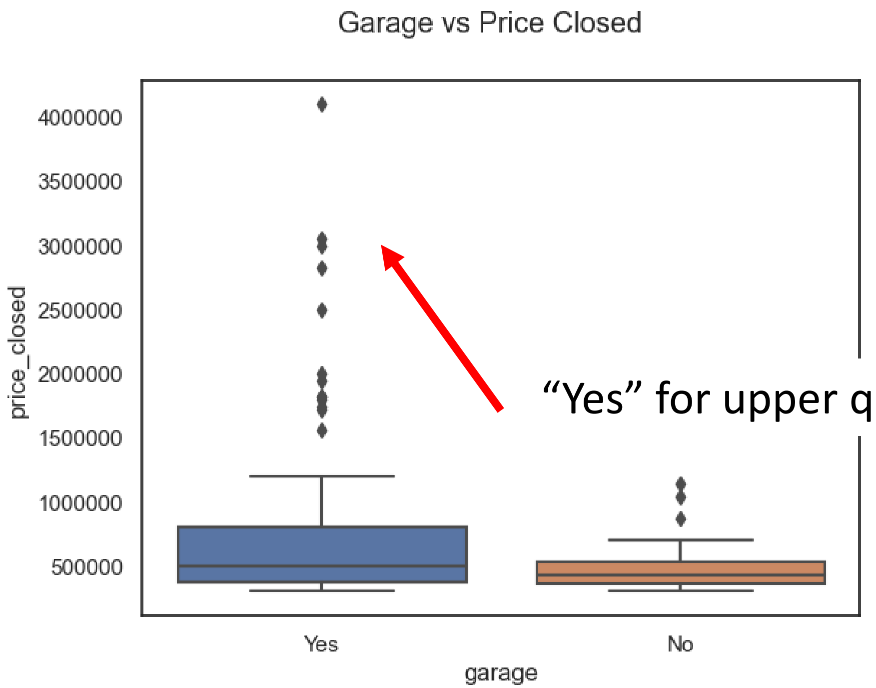
https://www.zillow.com/home-values/102001/united-states/



Similar across all types



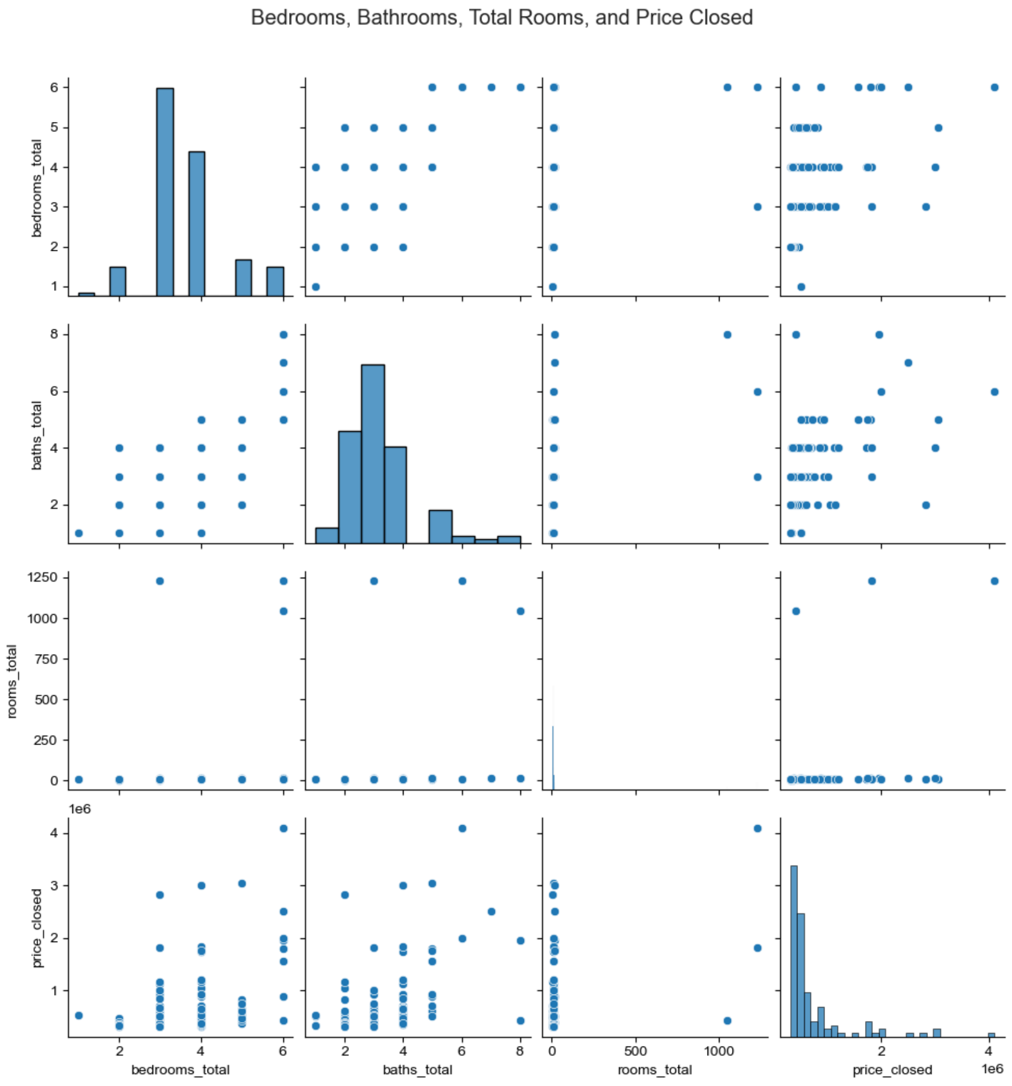
Peaks at 2 car garage



“Yes” for upper quartile

“Yes” for upper quartile

Rooms – bedrooms, bathrooms, total rooms




```
df['bedrooms_total'].describe()

count    122.000000
mean       3.598361
std        0.993032
min        1.000000
25%        3.000000
50%        3.000000
75%        4.000000
max        6.000000
Name: bedrooms_total, dtype: float64
```

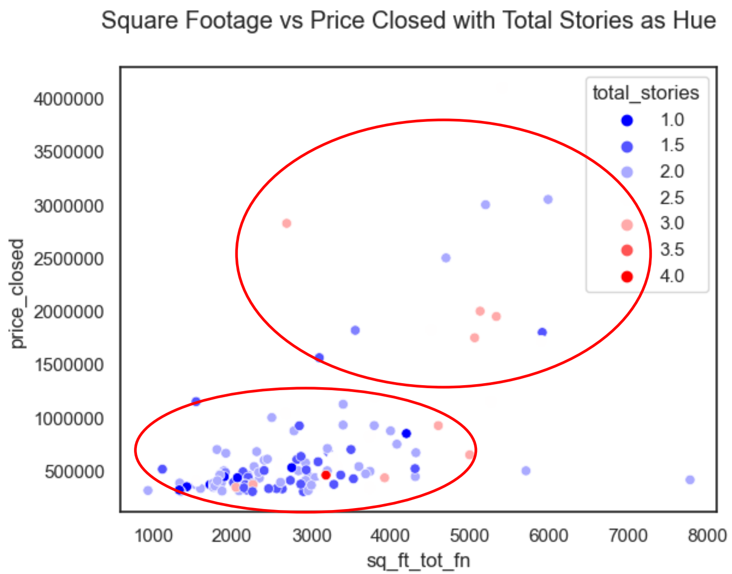
```
df['rooms_total'].describe()

count    122.000000
mean       9.196721
std        2.471789
min        5.000000
25%        7.000000
50%        9.000000
75%       10.000000
max       19.000000
Name: rooms_total, dtype: float64
```

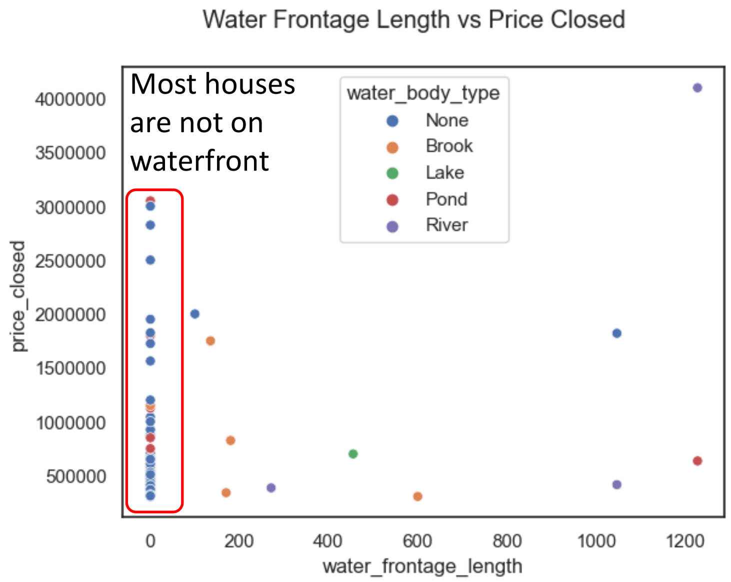
- Houses with more bedrooms and bathrooms have higher prices.
- While bathrooms has a peak at 6 rooms, both bedrooms and total rooms have peaks at the highest value.
- What is total rooms?
- Half baths?

No correlation if outlier is kept

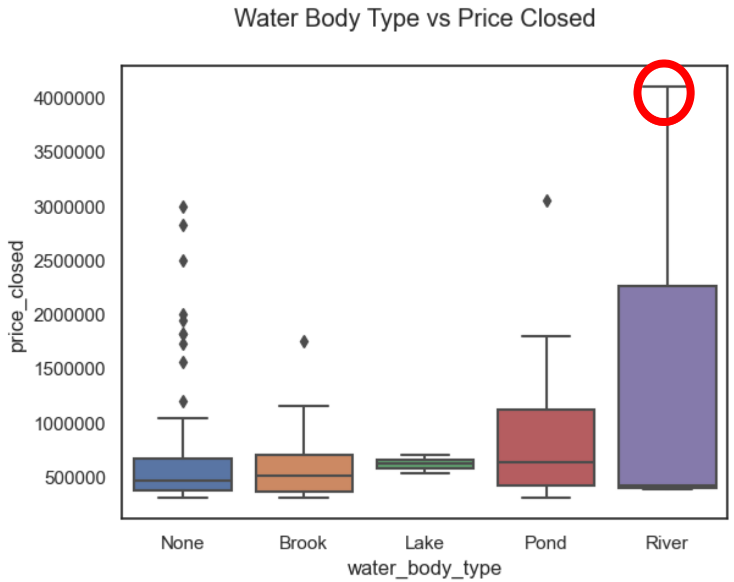
Positive correlation without outlier



Positive correlation between x and y
Darker at lower sq footage and lighter at higher sq footage

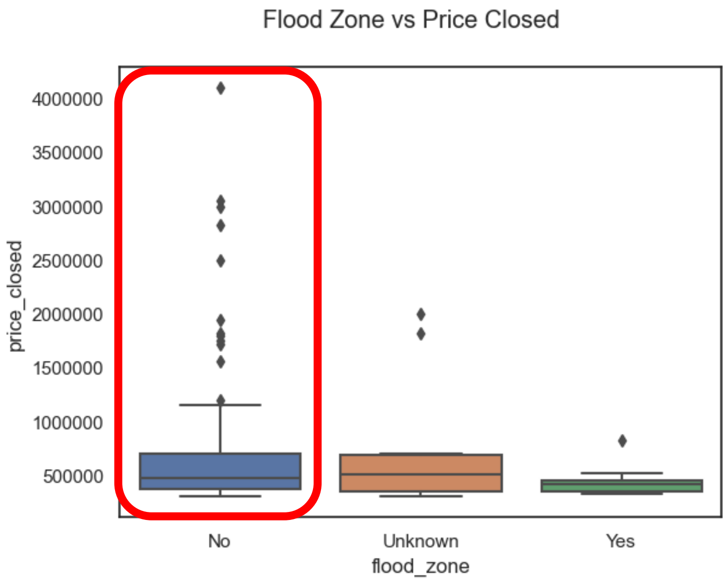


Water frontage length has no correlation with price closed or type

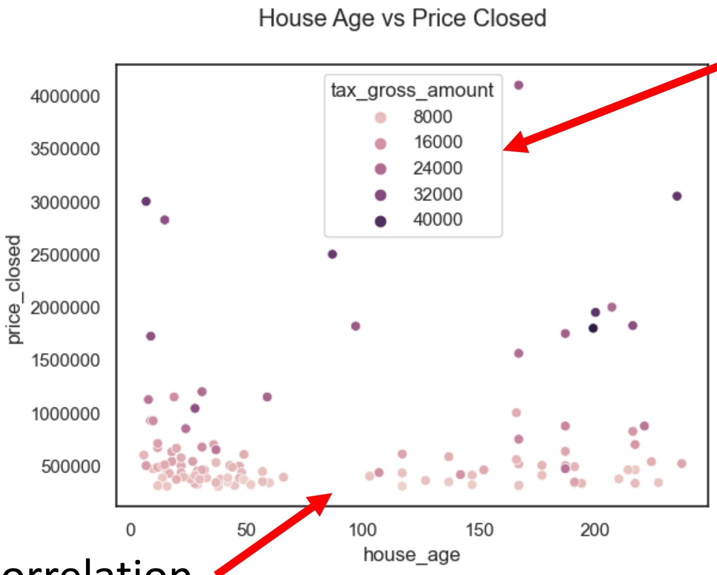


River has lowest median yet largest upper quartiles – due to outliers

Varying prices among the water bodies and no clear pattern

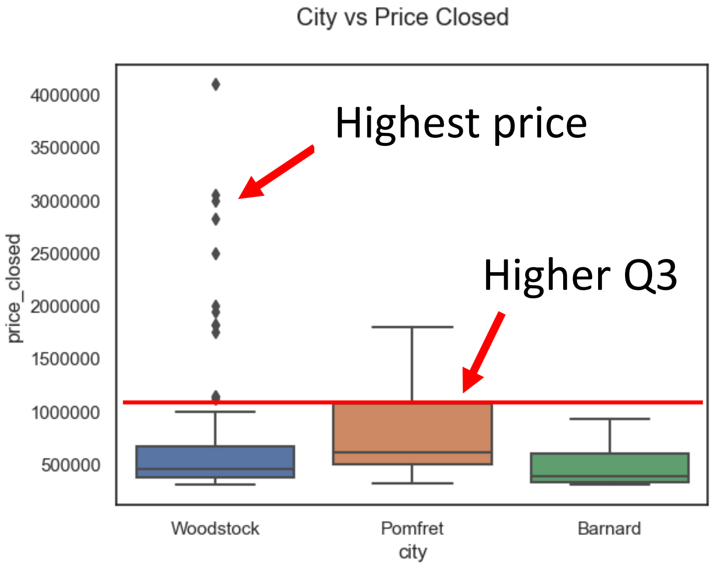


No flood zone has highest Q3 and contains upper outliers



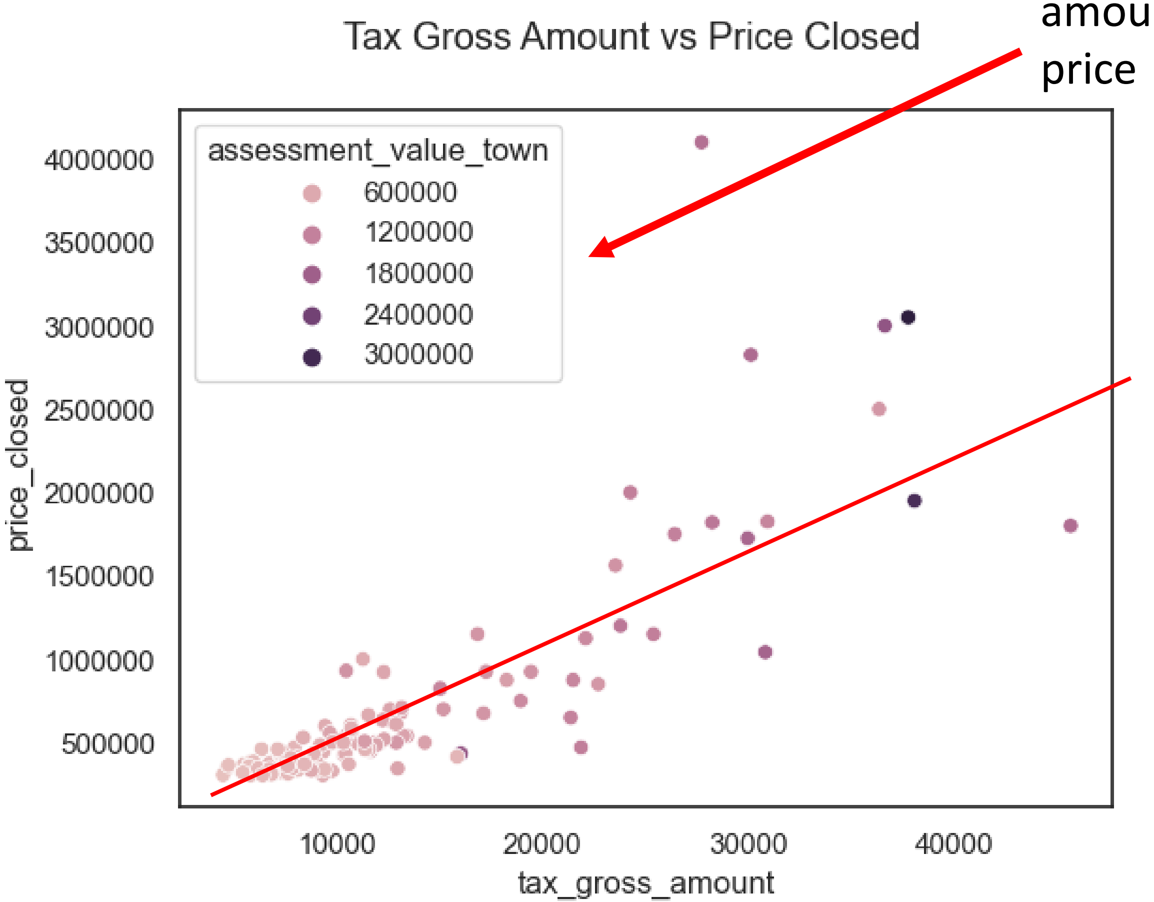
Tax gross amount increases with price closed but not house age

No correlation



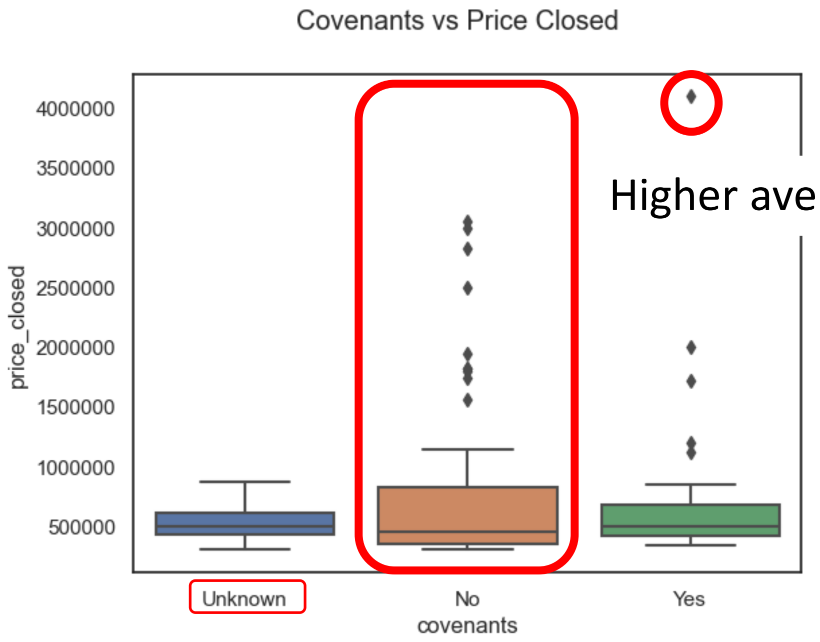
Highest price

Higher Q3

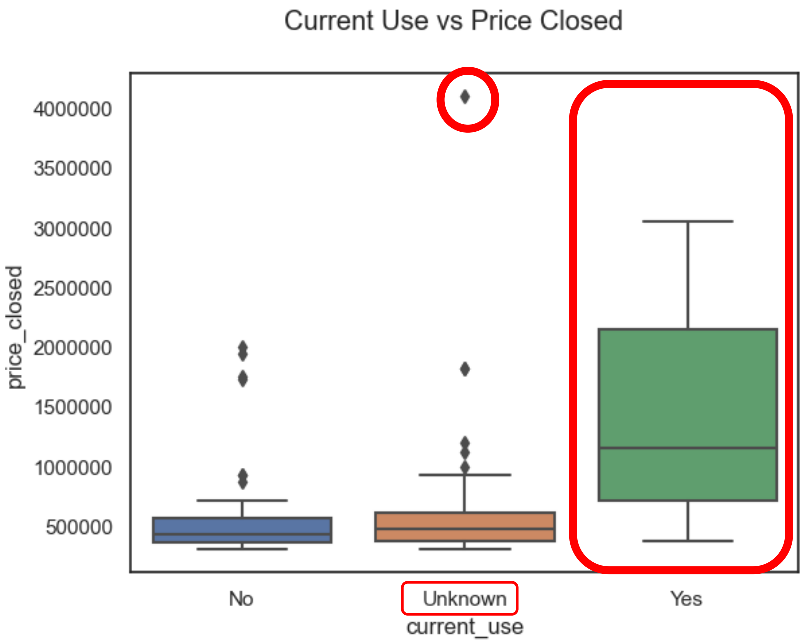


Assessment value increases with tax amount and price

Positive correlation between tax amount and price

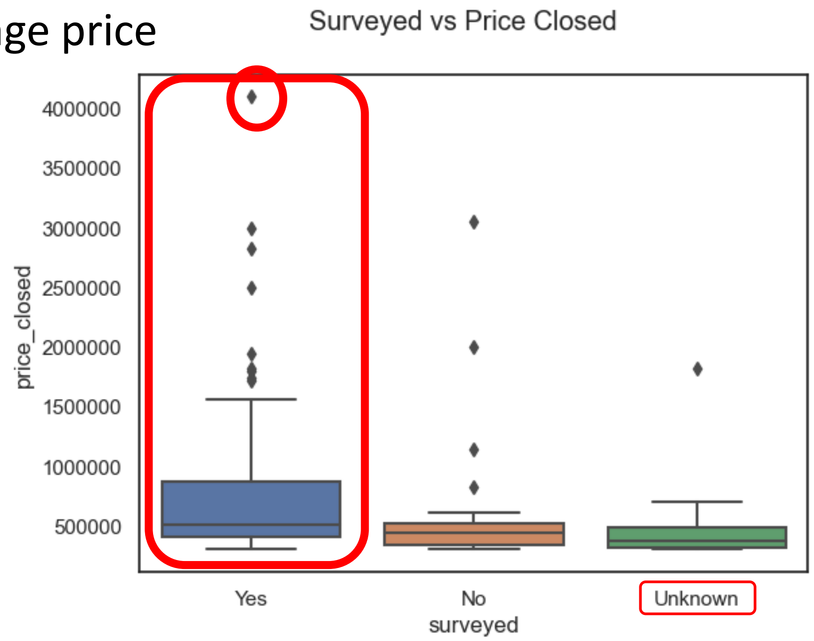


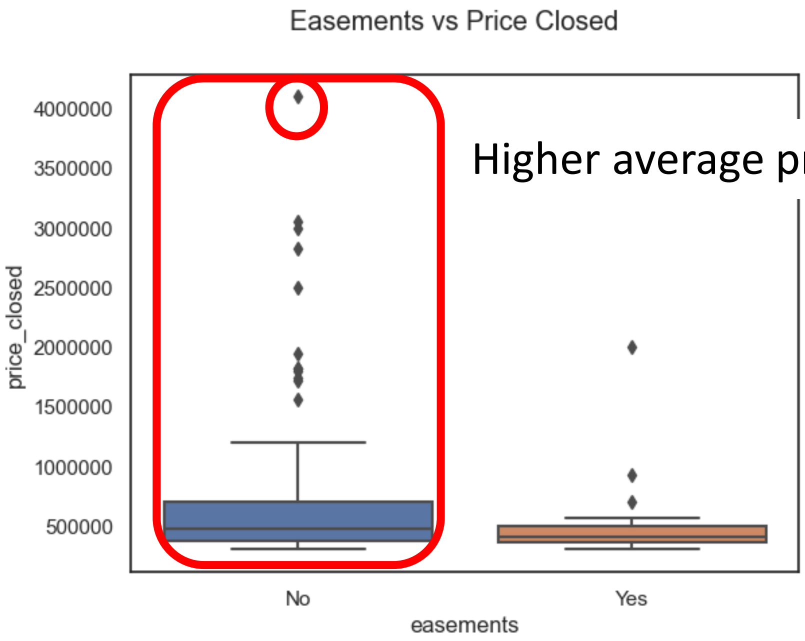
Higher average price



Higher average price

Higher average price





Higher average price

Hypothesis/Patterns/Anomalies

2 main hypotheses:

1. The columns with the most ‘unknown’ values will be the least useful in predicting prices.
2. City is most important.

Patterns:

- Houses at the lower end of the x variables are similar in price, while houses at the extreme values of these variables have more varied and higher prices.
- Houses with values ‘Unknown’ are priced similarly to houses with ‘no’ for variables such as current use and surveyed.

Anomalies:

- Garage and type of garage have weak correlations.
- Another anomaly is that house age has a weak correlation to price closed.

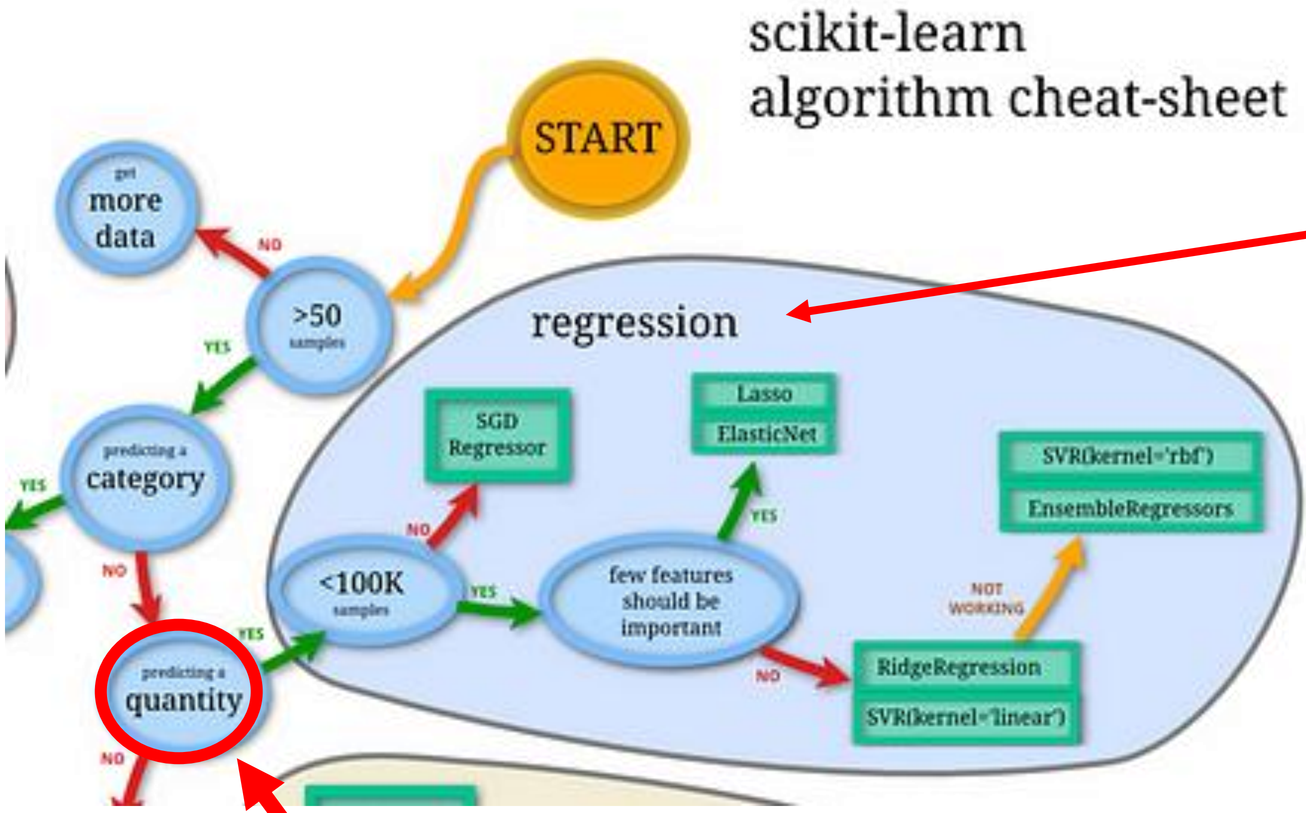
Outliers:

- There are 13 outliers in the upper range of the prices. These houses had a price closed of more than $1,316,250. These points could be the reason why the average house in this real estate market is significantly more expected than the average house in the US.

Problems With Data and Improvements

- The main problems with this dataset were present in its format
  - Lots of missing values, which translated into many ‘Unknown’ values in the cleaned dataset
  - Needed to convert values to numbers to successfully correlate
- Other problems were present in the correlations themselves
  - Price anomalies skewed some correlations
  - Houses seemed to have little correlation with some of our expected categories
    - Location relative to water in particular
- There are several ways to improve this dataset centering around getting a more representative sample size
  - Include more cities in and around current area
  - Expand housing criteria to include more residential areas that would be more indicative of average consumer
  - Choose a state with a real estate market more typical of the average American home.

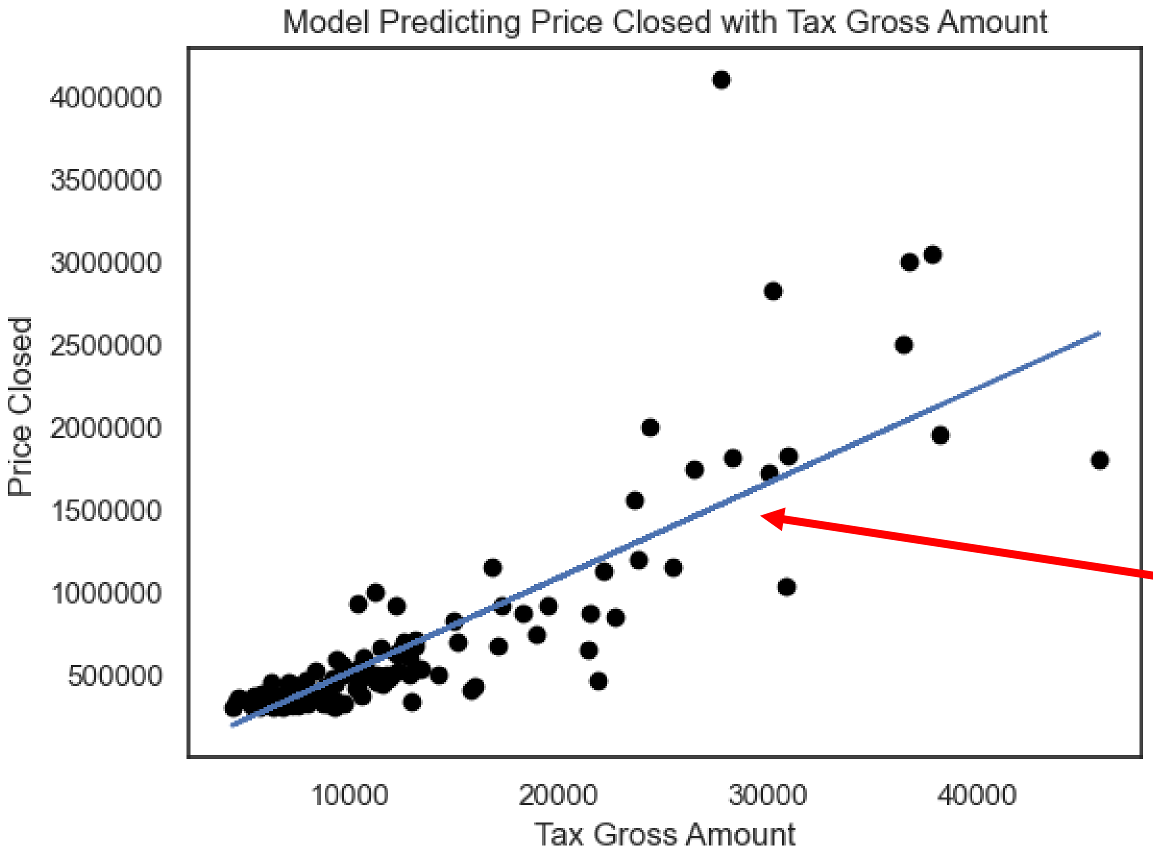
Machine Learning Model Chosen



We chose regression because we are predicting a value (the closing price).

Most influential variable: tax gross amount

```
X_train_tga, X_test_tga, Y_train_tga, Y_test_tga =
train_test_split(df[['tax_gross_amount']], df[['price_closed']], test_size=0.3, random_state=4)
house_tga_reg = LinearRegression()
house_tga_reg.fit(X_train_tga, Y_train_tga)
```



Metrics To Evaluate Machine Learning model

```
180478.892770154
[31.36323151]
Mean Absolute Error: 74820.544553494
Mean Squared Error: 7737048542.364685
Root Mean Squared Error: 87960.49421396338
Variance score: 0.82
```

Tax gross amount had the highest correlation of 0.840

Adding numeric variables

Variables included	Variance
Tax gross amount	0.82
+ assessment value town	0.82
+ total stories	0.79
+ assessment value town, rooms total	0.78
+ assessment value town, baths total	0.81
+ tax gross amount, assessment value town, square footage	0.80

The best model

```
Mean Squared Error: 7737048542.364688
```

No change in variance but higher MSE, even though the correlation was 0.758

```
Mean Squared Error: 7737474850.672815
```

Variables with higher correlations lower the variance score

Feature Engineering

Variables included	**Variance**
Tax gross amount	0.82
+ assessment value town, city	0.83

```python
df = pd.concat([df, pd.get_dummies(df['city'], prefix = 'city')], axis=1)
```

```python
df.drop(['city'], axis=1, inplace=True)
```

+ assessment value town, city, current use	0.82

```python
df = pd.concat([df, pd.get_dummies(df['current_use'], prefix = 'current_use')], axis=1)
df.drop(['current_use'], axis=1, inplace=True)
```

```
df.info()

0   id                      109 non-null   int64
1   bedrooms_total          109 non-null   int64
2   baths_total             109 non-null   int64
3   acres                   109 non-null   float64
4   sq_ft_tot_fn            109 non-null   int64
5   tax_gross_amount        109 non-null   float64
6   assessment_value_town   109 non-null   float64
7   garage_capacity         109 non-null   float64
8   address                 109 non-null   object
9   garage_type             109 non-null   object
10  year_built              109 non-null   int64
11  total_stories           109 non-null   float64
12  surveyed                109 non-null   object
13  water_body_type         109 non-null   object
14  water_frontage_length   109 non-null   int64
15  rooms_total             109 non-null   int64
16  garage                  109 non-null   object
17  flood_zone              109 non-null   object
18  easements               109 non-null   object
19  covenants               109 non-null   object
20  basement_access_type    109 non-null   object
21  basement                109 non-null   object
22  price_closed            109 non-null   int64
23  house_age               109 non-null   int64
24  city_Barnard            109 non-null   uint8
25  city_Pomfret            109 non-null   uint8
26  city_Woodstock          109 non-null   uint8
27  current_use_No          109 non-null   uint8
28  current_use_Unknown     109 non-null   uint8
29  current_use_Yes         109 non-null   uint8
```

Best model with metrics

Tax Gross Amount and City

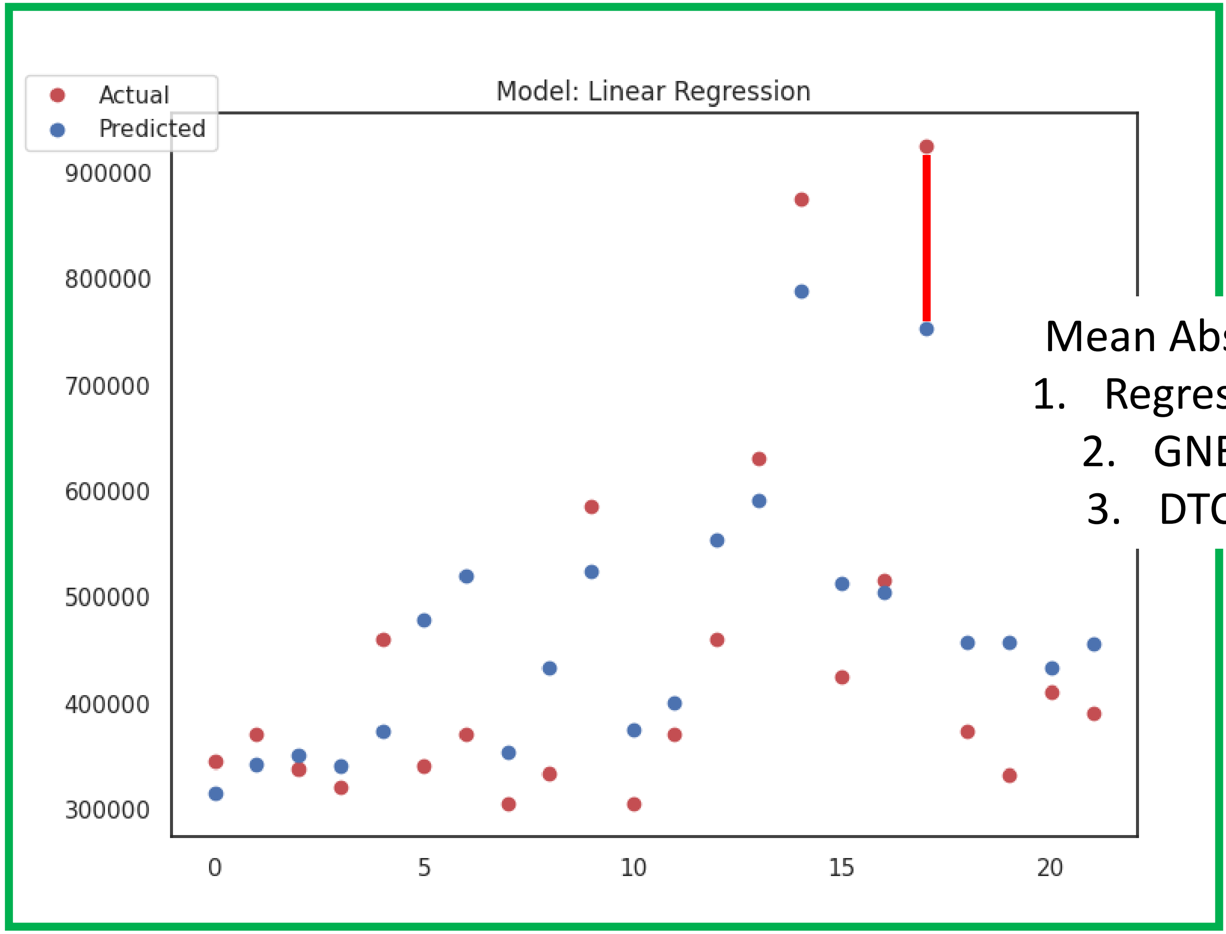
```
X = df.iloc[:, [5,24,25,26]].values
y = df.iloc[:, 22].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=4)
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
df_regressor = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
print(df_regressor)
print(model.intercept_)
print(model.coef_)
predictions = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
pd.set_option('display.float_format', '{:.2f}'.format)
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
print('Variance score: %.2f' % model.score(X_test, y_test))
```

	Actual	Predicted
0	345000	322400.20
1	370000	352514.01
2	338000	359362.91
3	320000	350870.13
4	460000	381021.80
5	340000	481642.24
6	370000	521553.75
7	305000	363017.00
8	334000	438091.09
9	585000	525650.17
10	305000	382858.67
28	1150000	1013237.81

```
Mean Absolute Error: 73751.86117077102
Mean Squared Error: 7305542731.840339
Root Mean Squared Error: 85472.46768311033
Variance score: 0.83
```

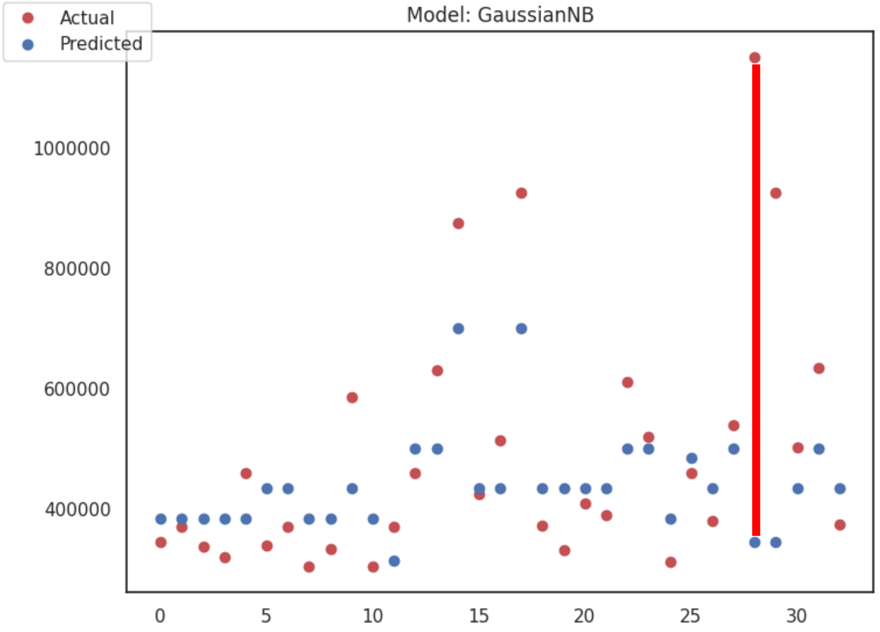
Differences can range from <2,000 to >10,000 in the lower ranges, but the model becomes less accurate at higher house prices.

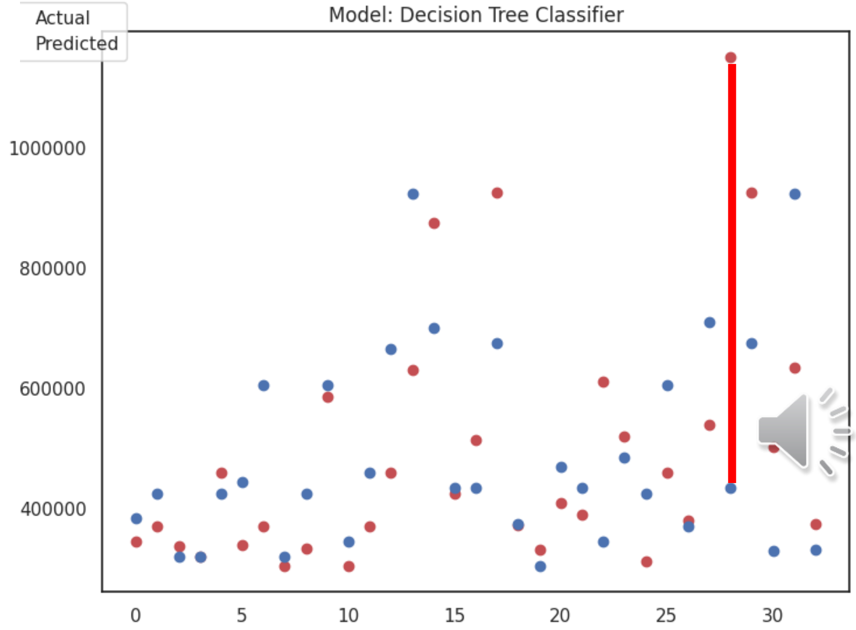
Comparing Models



Mean Absolute Error:

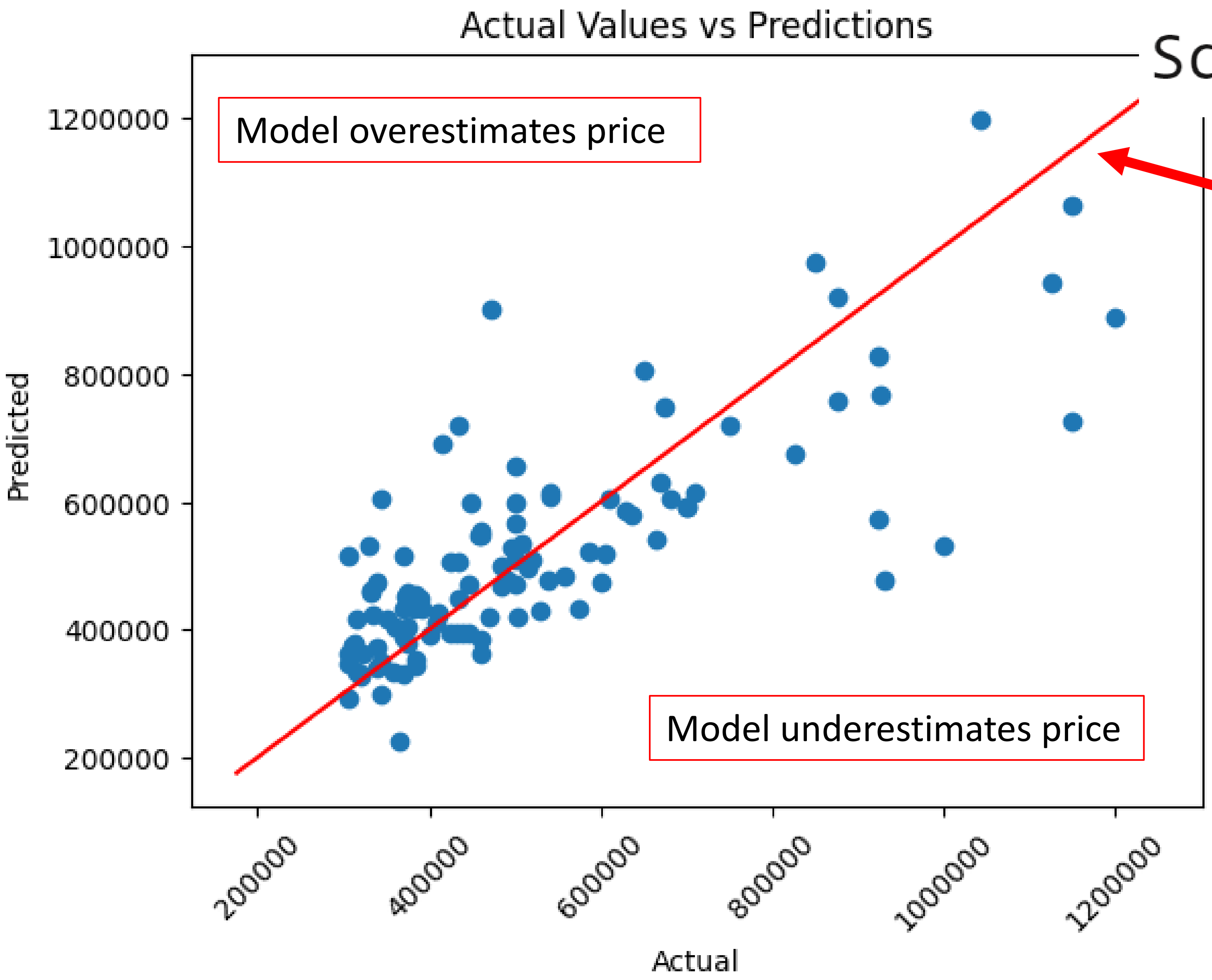
1. Regression (73752)
2. GNB (111600)
3. DTC (124322)





Cross Validation

```
scores = cross_val_score(model, x_values, y_values, cv=6)
print('Cross-validated scores:', scores)
```



Score: 0.7451762889767538

This 45-degree line shows where the predicted value is equal to the actual value.

The error increases at higher house prices, and the model tends to underestimate the prices of more expensive homes.

Conclusion

- **Our hypothesis was that the city would be the most influential in predicting the closing price of the house, and that characteristics with lots of unknown data points would be the least influential.**
  - Interestingly, tax gross amount turned out to be the most influential factor on closing price. The property locations had an identical variance value (0.82) to tax gross amount, yet taxes had a lower mean error indicating a stronger correlation.
  - We found that the unknown categories were less influential in determining the closing price, which makes sense given that there was less overall data to work with in those categories.
  - Our successfully established correlation between tax gross amount and closing price indicates that income might be a more prevalent factor in determining the closing price of the house in the state of Vermont
- **The Dataset and Machine Learning**
  - The dataset itself was fairly large, however it required lots of cleaning in order to make it suitable for machine learning
  - Lots of unknown values in various categories as well as obvious outliers as prices went into the millions
  - We found that a regression model best suited the data set and used Gaussian and a Decision Tree Classifier in order to compare and cross validate
  - The model showed a strong positive correlation between tax gross amount and price closed, however it began to deteriorate and underestimate the price closed at higher values
  - A regression model was best suited for our project because we were predicting a fixed value of closing price based on other outside factors

- **Future Efforts**
  - The dataset presented initially with data from a few cities around Vermont representative of the state's housing market
  - We did not expand on this data set by bringing in outside data, instead we cleaned the data to make it suitable for a regression model
  - Adding more cities from Vermont in order to make a more representative sample of American suburbs seems like a logical next step to expand our investigation
  - Our investigation also hinted that income (as a consequence of tax gross amount) might be an additional factor that affects house closing prices
  - Adding data that includes median family incomes might prove useful in further research.
- **Other Research**
- A 'Vermont perfect storm': Statewide data shows record spike in housing prices – VTDigger – 2023 Article by VTDigger
- "Extremely low interest rates and Vermont's relatively low number of Covid-19 cases pumped up demand in the state's housing market from out-of-state homebuyers, Palmer said. They were motivated not just by relative safety from the virus, Palmer said, but also because of the realization that working remotely was possible. "