## 0.1 Question 1: Human Context and Ethics

In this part of the project, we will explore the human context of our housing dataset.

**You should read the Project_CaseStudy.pdf on Canvas explaining the context and history surrounding this dataset before attempting this section.**

---

### 0.1.1 Question 1a

"How much is a house worth?" Who might be interested in an answer to this question? **Please list at least three different parties (people or organizations) and state whether each one has an interest in seeing the housing price to be high or low.**

Three examples of parties who might be interested in how much a house is worth are: Buyers, Sellers, and Members Of The Community of where the property may or may not be sold.

- **Buyers**: A buyer in this context could have mixed feelings about what interest they have in regards to the price of a house. More often than not, buyers are going to be hopeful that a house price is lower than it is higher.
- **Sellers**: Sellers are obviously going to want the housing price to be higher rather than lower. This is because they are more than likely wanting to gain as much capital from a sale as possible.
- **Members Of The Community**: Members of the community (like neighbors) are more than likely going to want housing prices to be higher than lower so that the homes that they own, are also worth more than less.

### 0.1.2 Question 1b

Which of the following scenarios strike you as unfair and why? You can choose more than one. There is no single right answer, but you must explain your reasoning. Would you consider some of these scenarios more (or less) fair than others? Why?

A. A homeowner whose home is assessed at a higher price than it would sell for.
B. A homeowner whose home is assessed at a lower price than it would sell for.
C. An assessment process that systematically overvalues inexpensive properties and undervalues expensive properties.
D. An assessment process that systematically undervalues inexpensive properties and overvalues expensive properties.

I first find scenario **B** to be unfair. If the price of the house is assessed as lower than what it would sell for, the homeowner could potentially lose money on the sale of the house. I find scenarios **C** and **D** to be both unfair because this process is systematically screwing over either buyers or sellers of the home. In one context, buyers may be purchasing a home for more than it is worth and in the other sellers may be getting less than what their home is worth. Both in my eyes are unfair.

### 0.1.3 Question 1d

What were the central problems with the earlier property tax system in Cook County as reported by the Chicago Tribune ? And what were the primary causes of these problems? (Note: in addition to reading the paragraph above you will need to **read the Project_CaseStudy.pdf explaining the context and history of this dataset before answering this question).**

The Chicago Tribune reported that there was a regressive tax system for home values. Essentially, homes that were evaluated at a lower price were taxed at a higher rate than those that were evaluated at a higher price. This caused a problem because those who had a lower house evaluation were paying more in taxes (as a ratio) than those who had a higher house evaluation.

## 0.2 Question 2a: More EDA

In good news you have already done a lot of EDA with this dataset in Project 1.

Before fitting any model, we should check for any missing data and/or unusual outliers.

Since we're trying to predict `Sale Price`, we'll start with that field.

Examine the `Sale Price` column in the `training_val_data` DataFrame and answer the following questions:

- 2ai). Does the `Sale Price` data have any missing, N/A, negative or 0 values for the data? If so, propose a way to handle this.
- 2aii). Does the `Sale Price` data have any unusually large outlier values? If so, propose a cutoff to use for throwing out large outliers, and justify your reasoning).
- 2aiii). Does the `Sale Price` data have any unusually small outlier values? If so, propose a cutoff to use for throwing out small outliers, and justify your reasoning.

Below are three cells. The first is a Markdown cell for you to write up your responses to all 3 parts above. The second two are code cells that are available for you to write code to explore the outliers and/or visualize the Sale Price data.

### 0.2.1 Question 2abc answer cell:**

**Part 2ai** From my inspection, `Sale Price` does not contain any N/A, negative, or 0 values for the data. If it did, we could filter the data frame to remove these values.

**Part 2aii** Yes, `Sale Price` does have some large outliers in it. In fact, from my inspection, there are 33 homes in this data set that are or above the cutoff of $5,000,000. These are extreme outliers. We could filter the data in this set so that these outliers were removed, similar to if we had values that were invalid.

**Part 2aiii** Similar to part ii of this problem, there are a wide number of sale prices in the `Sale Price` column that are at or below $500. Because these could create problems in the data set, we could filter them out so their effects were not felt as hard. Similar to part ii for the high outliers.

```
In [9]: print(training_val_data[training_val_data['Sale Price'] >= 5000000])
        # your code exploring Sale Price above this line
```

| | PIN | Property Class | Neighborhood Code | Land Square Feet | \ |
|---|---|---|---|---|---|
| 929 | 5214120120000 | 209 | 171 | 58500.000000 | |
| 3838 | 14333031660000 | 209 | 12 | 4471.000000 | |
| 4657 | 17042100070000 | 209 | 22 | 3762.000000 | |
| 7619 | 14333020690000 | 209 | 12 | 4092.000000 | |
| 8043 | 9351120170000 | 278 | 150 | 5850.000000 | |
| 10657 | 14333020330000 | 208 | 12 | 4092.000000 | |
| 10961 | 5274140030000 | 206 | 171 | 29100.000000 | |
| 19702 | 5172030380000 | 209 | 171 | 46174.000000 | |
| 30684 | 14331230470000 | 206 | 12 | 6800.000000 | |
| 37835 | 5214030240000 | 209 | 171 | 51156.000000 | |
| 65436 | 5083030290000 | 209 | 171 | 38077.000000 | |
| 68886 | 5274040100000 | 204 | 171 | 26500.000000 | |
| 71953 | 5173130050000 | 209 | 80 | 74052.000000 | |
| 74846 | 5064040400000 | 209 | 171 | 46000.000000 | |
| 76367 | 5161060720000 | 209 | 171 | 28218.000000 | |
| 79388 | 5214120100000 | 204 | 171 | 47226.000000 | |
| 92407 | 5084000450000 | 204 | 171 | 39735.000000 | |
| 97707 | 14333031630000 | 209 | 12 | 4100.000000 | |
| 100520 | 5291000560000 | 209 | 80 | 104596.000000 | |
| 106057 | 19284170370000 | 203 | 50 | 4687.000000 | |
| 117989 | 17031000110000 | 209 | 22 | 5500.000000 | |
| 121800 | 5211140020000 | 209 | 171 | 38507.000000 | |
| 128577 | 14333021670000 | 209 | 12 | 3683.000000 | |
| 131247 | 5161060440000 | 209 | 171 | 27600.000000 | |
| 144568 | 5081010340000 | 209 | 171 | 39438.000000 | |
| 153098 | 5274040090000 | 209 | 171 | 26800.000000 | |
| 157521 | 5161060740000 | 209 | 171 | 71715.685936 | |
| 159016 | 5081000040000 | 209 | 170 | 32736.000000 | |
| 168026 | 17031000020000 | 209 | 22 | 3230.000000 | |
| 172491 | 27024050050000 | 204 | 32 | 10500.000000 | |
| 188753 | 5204070110000 | 209 | 80 | 55060.000000 | |
| 193635 | 5211040010000 | 209 | 171 | 62962.705122 | |
| 194905 | 5081010510000 | 209 | 171 | 41948.000000 | |

| | Town Code | Apartments | Wall Material | Roof Material | Basement | \ |
|---|---|---|---|---|---|---|
| 929 | 23 | 0.0 | 2.0 | 5.0 | 1.0 | |
| 3838 | 74 | 0.0 | 2.0 | 2.0 | 1.0 | |
| 4657 | 74 | 0.0 | 2.0 | 2.0 | 1.0 | |
| 7619 | 74 | 0.0 | 2.0 | 2.0 | 1.0 | |
| 8043 | 22 | 0.0 | 2.0 | 1.0 | 1.0 | |
| 10657 | 74 | 0.0 | 2.0 | 2.0 | 1.0 | |
| 10961 | 23 | 0.0 | 2.0 | 5.0 | 1.0 | |
| 19702 | 23 | 0.0 | 2.0 | 3.0 | 1.0 | |
| 30684 | 74 | 0.0 | 2.0 | 2.0 | 1.0 | |
| 37835 | 23 | 0.0 | 2.0 | 3.0 | 1.0 | |
| 65436 | 23 | 0.0 | 1.0 | 2.0 | 3.0 | |
| 68886 | 23 | 0.0 | 3.0 | 1.0 | 1.0 | |
| 71953 | 23 | 0.0 | 2.0 | 3.0 | 1.0 | |
| 74846 | 23 | 0.0 | 2.0 | 6.0 | 3.0 | |
| 76367 | 23 | 0.0 | 2.0 | 3.0 | 1.0 | |
| 79388 | 23 | 0.0 | 2.0 | 3.0 | 3.0 | |
| 92407 | 23 | 0.0 | 2.0 | 3.0 | 1.0 | |
| 97707 | 74 | 0.0 | 2.0 | 2.0 | 1.0 | |

| | | | | | |
|---|---|---|---|---|---|
| 100520 | 23 | 0.0 | 2.0 | 1.0 | 3.0 |
| 106057 | 36 | 0.0 | 2.0 | 1.0 | 1.0 |
| 117989 | 74 | 0.0 | 2.0 | 6.0 | 3.0 |
| 121800 | 23 | 0.0 | 2.0 | 4.0 | 1.0 |
| 128577 | 74 | 0.0 | 2.0 | 1.0 | 1.0 |
| 131247 | 23 | 0.0 | 2.0 | 4.0 | 1.0 |
| 144568 | 23 | 0.0 | 3.0 | 2.0 | 3.0 |
| 153098 | 23 | 0.0 | 2.0 | 3.0 | 1.0 |
| 157521 | 23 | 0.0 | 2.0 | 5.0 | 1.0 |
| 159016 | 23 | 0.0 | 2.0 | 5.0 | 3.0 |
| 168026 | 74 | 0.0 | 2.0 | 6.0 | 1.0 |
| 172491 | 28 | 0.0 | 2.0 | 1.0 | 3.0 |
| 188753 | 23 | 0.0 | 2.0 | 5.0 | 1.0 |
| 193635 | 23 | 0.0 | 3.0 | 5.0 | 1.0 |
| 194905 | 23 | 0.0 | 2.0 | 3.0 | 1.0 |

| | Basement Finish | … | Sale Month of Year | Sale Half of Year | \ |
|---|---|---|---|---|---|
| 929 | 1.0 | … | 1 | 1 | |
| 3838 | 1.0 | … | 10 | 2 | |
| 4657 | 1.0 | … | 6 | 1 | |
| 7619 | 1.0 | … | 1 | 1 | |
| 8043 | 3.0 | … | 10 | 2 | |
| 10657 | 1.0 | … | 9 | 2 | |
| 10961 | 3.0 | … | 9 | 2 | |
| 19702 | 1.0 | … | 5 | 1 | |
| 30684 | 1.0 | … | 5 | 1 | |
| 37835 | 1.0 | … | 7 | 2 | |
| 65436 | 1.0 | … | 4 | 1 | |
| 68886 | 1.0 | … | 4 | 1 | |
| 71953 | 3.0 | … | 10 | 2 | |
| 74846 | 1.0 | … | 11 | 2 | |
| 76367 | 3.0 | … | 1 | 1 | |
| 79388 | 3.0 | … | 6 | 1 | |
| 92407 | 1.0 | … | 6 | 1 | |
| 97707 | 1.0 | … | 4 | 1 | |
| 100520 | 3.0 | … | 11 | 2 | |
| 106057 | 1.0 | … | 3 | 1 | |
| 117989 | 1.0 | … | 8 | 2 | |
| 121800 | 1.0 | … | 8 | 2 | |
| 128577 | 1.0 | … | 5 | 1 | |
| 131247 | 3.0 | … | 12 | 2 | |
| 144568 | 1.0 | … | 4 | 1 | |
| 153098 | 3.0 | … | 2 | 1 | |
| 157521 | 3.0 | … | 1 | 1 | |
| 159016 | 3.0 | … | 3 | 1 | |
| 168026 | 1.0 | … | 1 | 1 | |
| 172491 | 3.0 | … | 4 | 1 | |
| 188753 | 1.0 | … | 5 | 1 | |
| 193635 | 3.0 | … | 5 | 1 | |
| 194905 | 1.0 | … | 12 | 2 | |

| | Most Recent Sale | Age Decade | Pure Market Filter | Garage Indicator | \ |
|---|---|---|---|---|---|
| 929 | 1.0 | 1.5 | 0 | 1.0 | |
| 3838 | 1.0 | 0.1 | 1 | 1.0 | |

| | | | | |
|---|---|---|---|---|
| 4657 | 1.0 | 13.0 | 1 | 1.0 |
| 7619 | 1.0 | 0.1 | 1 | 1.0 |
| 8043 | 0.0 | 1.7 | 1 | 1.0 |
| 10657 | 1.0 | 2.4 | 1 | 1.0 |
| 10961 | 1.0 | 9.8 | 1 | 1.0 |
| 19702 | 1.0 | 0.1 | 0 | 1.0 |
| 30684 | 1.0 | 7.9 | 1 | 1.0 |
| 37835 | 1.0 | 0.7 | 0 | 1.0 |
| 65436 | 1.0 | 2.2 | 1 | 1.0 |
| 68886 | 1.0 | 6.2 | 1 | 1.0 |
| 71953 | 1.0 | 8.8 | 0 | 1.0 |
| 74846 | 1.0 | 1.5 | 1 | 1.0 |
| 76367 | 1.0 | 1.8 | 1 | 1.0 |
| 79388 | 1.0 | 8.6 | 1 | 1.0 |
| 92407 | 1.0 | 2.1 | 1 | 1.0 |
| 97707 | 1.0 | 0.1 | 1 | 1.0 |
| 100520 | 1.0 | 7.7 | 1 | 0.0 |
| 106057 | 0.0 | 4.8 | 0 | 1.0 |
| 117989 | 1.0 | 11.7 | 1 | 1.0 |
| 121800 | 1.0 | 5.1 | 1 | 1.0 |
| 128577 | 1.0 | 1.2 | 1 | 1.0 |
| 131247 | 1.0 | 1.3 | 1 | 0.0 |
| 144568 | 1.0 | 1.0 | 1 | 1.0 |
| 153098 | 1.0 | 9.2 | 1 | 1.0 |
| 157521 | 1.0 | 10.9 | 1 | 0.0 |
| 159016 | 1.0 | 10.4 | 1 | 1.0 |
| 168026 | 0.0 | 11.7 | 1 | 1.0 |
| 172491 | 0.0 | 4.0 | 1 | 1.0 |
| 188753 | 1.0 | 0.1 | 1 | 1.0 |
| 193635 | 0.0 | 8.8 | 1 | 0.0 |
| 194905 | 1.0 | 0.7 | 1 | 1.0 |

| | Neigborhood Code (mapping) | Town and Neighborhood \ |
|---|---|---|
| 929 | 171 | 23171 |
| 3838 | 12 | 7412 |
| 4657 | 22 | 7422 |
| 7619 | 12 | 7412 |
| 8043 | 150 | 22150 |
| 10657 | 12 | 7412 |
| 10961 | 171 | 23171 |
| 19702 | 171 | 23171 |
| 30684 | 12 | 7412 |
| 37835 | 171 | 23171 |
| 65436 | 171 | 23171 |
| 68886 | 171 | 23171 |
| 71953 | 80 | 2380 |
| 74846 | 171 | 23171 |
| 76367 | 171 | 23171 |
| 79388 | 171 | 23171 |
| 92407 | 171 | 23171 |
| 97707 | 12 | 7412 |
| 100520 | 80 | 2380 |
| 106057 | 50 | 3650 |
| 117989 | 22 | 7422 |

```
121800                        171               23171
128577                         12                7412
131247                        171               23171
144568                        171               23171
153098                        171               23171
157521                        171               23171
159016                        170               23170
168026                         22                7422
172491                         32                2832
188753                         80                2380
193635                        171               23171
194905                        171               23171

                                                   Description       Lot Size
929     This property, sold on 01/10/2013, is a two-st…   58500.000000
3838    This property, sold on 10/04/2019, is a three-…    4471.000000
4657    This property, sold on 06/19/2018, is a three-…    3762.000000
7619    This property, sold on 01/11/2016, is a three-…    4092.000000
8043    This property, sold on 10/07/2015, is a two-st…    5850.000000
10657   This property, sold on 09/17/2018, is a three-…    4092.000000
10961   This property, sold on 09/19/2013, is a two-st…   29100.000000
19702   This property, sold on 05/07/2018, is a two-st…   46174.000000
30684   This property, sold on 05/20/2014, is a two-st…    6800.000000
37835   This property, sold on 07/17/2013, is a two-st…   51156.000000
65436   This property, sold on 04/12/2019, is a two-st…   38077.000000
68886   This property, sold on 04/03/2015, is a one-st…   26500.000000
71953   This property, sold on 10/26/2015, is a two-st…   74052.000000
74846   This property, sold on 11/04/2014, is a two-st…   46000.000000
76367   This property, sold on 01/05/2016, is a three-…   28218.000000
79388   This property, sold on 06/15/2015, is a one-st…   47226.000000
92407   This property, sold on 06/27/2017, is a one-st…   39735.000000
97707   This property, sold on 04/23/2018, is a three-…    4100.000000
100520  This property, sold on 11/14/2013, is a two-st…  104596.000000
106057  This property, sold on 03/25/2013, is a one-st…    4687.000000
117989  This property, sold on 08/19/2014, is a three-…    5500.000000
121800  This property, sold on 08/31/2016, is a two-st…   38507.000000
128577  This property, sold on 05/24/2018, is a three-…    3683.000000
131247  This property, sold on 12/28/2016, is a two-st…   27600.000000
144568  This property, sold on 04/18/2017, is a two-st…   39438.000000
153098  This property, sold on 02/06/2015, is a two-st…   26800.000000
157521  This property, sold on 01/10/2018, is a two-st…   71715.685936
159016  This property, sold on 03/05/2014, is a two-st…   32736.000000
168026  This property, sold on 01/22/2013, is a three-…    3230.000000
172491  This property, sold on 04/18/2017, is a one-st…   10500.000000
188753  This property, sold on 05/05/2014, is a two-st…   55060.000000
193635  This property, sold on 05/10/2013, is a two-st…   62962.705122
194905  This property, sold on 12/05/2013, is a two-st…   41948.000000

[33 rows x 62 columns]
```

In [10]:
```python
print(training_val_data[training_val_data['Sale Price'] <= 500])
# optional extra cell for exploring code
```

```
                  PIN  Property Class  Neighborhood Code  Land Square Feet  \
0       17294100610000             203                 50            2500.0
5       19174010300000             203                380            4390.0
14      21312250180000             202                100            2976.0
20       3323080010000             203                 70            6500.0
23       6171050140000             203                 11            9113.0
…                  …               …                 …                 …
204757  16264040290000             202                115            3125.0
204765  13174290110000             203                 90            4250.0
204766  25051260290000             205                282            4404.0
204773   3161170030000             278                 40            8750.0
204782  23293040180000             208                 52           29848.0

        Town Code  Apartments  Wall Material  Roof Material  Basement  \
0              76         0.0            2.0            1.0       1.0
5              72         0.0            2.0            1.0       1.0
14             70         0.0            1.0            1.0       1.0
20             38         0.0            2.0            1.0       1.0
23             18         0.0            1.0            1.0       1.0
…             …          …              …              …         …
204757         77         0.0            1.0            1.0       2.0
204765         71         0.0            2.0            1.0       1.0
204766         72         0.0            2.0            1.0       1.0
204773         38         0.0            1.0            1.0       3.0
204782         30         0.0            2.0            1.0       1.0

        Basement Finish  …  Sale Month of Year  Sale Half of Year  \
0                   3.0  …                   9                  2
5                   3.0  …                   7                  2
14                  3.0  …                   6                  1
20                  3.0  …                   2                  1
23                  3.0  …                   2                  1
…                   …   …                  …                  …
204757              3.0  …                   8                  2
204765              3.0  …                   4                  1
204766              3.0  …                   6                  1
204773              3.0  …                  11                  2
204782              3.0  …                   6                  1

        Most Recent Sale  Age Decade  Pure Market Filter  Garage Indicator  \
0                    1.0        13.2                   0               0.0
5                    1.0         5.8                   0               1.0
14                   0.0        12.2                   0               1.0
20                   1.0         9.4                   0               0.0
23                   0.0         4.1                   0               1.0
…                    …          …                  …                 …
204757               1.0        11.7                   0               1.0
204765               1.0         6.6                   0               1.0
204766               0.0         6.7                   0               1.0
204773               1.0         3.4                   0               1.0
204782               1.0         2.0                   0               1.0

        Neigborhood Code (mapping)  Town and Neighborhood  \
0                               50                   7650
```

```
5                                 380               72380
14                                100               70100
20                                 70                3870
23                                 11                1811
…                               …                 …
204757                            115               77115
204765                             90                7190
204766                            282               72282
204773                             40                3840
204782                             52                3052

                                              Description  Lot Size
0       This property, sold on 09/14/2015, is a one-st…    2500.0
5       This property, sold on 07/26/2018, is a one-st…    4390.0
14      This property, sold on 06/28/2017, is a one-st…    2976.0
20      This property, sold on 02/13/2019, is a one-st…    6500.0
23      This property, sold on 02/22/2019, is a one-st…    9113.0
…                                                   …       …
204757  This property, sold on 08/10/2017, is a one-st…    3125.0
204765  This property, sold on 04/18/2019, is a one-st…    4250.0
204766  This property, sold on 06/28/2016, is a two-st…    4404.0
204773  This property, sold on 11/18/2019, is a two-st…    8750.0
204782  This property, sold on 06/18/2019, is a two-st…   29848.0

[35873 rows x 62 columns]
```

## 0.3 Question 5: Improving the Model

### 0.3.1 Question 5a: Choose an additional feature

It's your turn to choose another feature to add to the model. Choose one new **quantitative** (not qualitative) feature and create Model 3 incorporating this feature (along with the features we've already chosen in Model 2). Try to choose a feature that will have a large impact on reducing the RMSE and/or will improve your residual plots. This can be a raw feature available in the dataset, or a transformation of one of the features in the dataset, or a new feature that you create from the dataset (see Project 1 for ideas). In the cell below, explain what additional feature you have chosen and why. Justify your reasoning. There are optional code cells provided below for you to use when exploring the dataset to determine which feature to add.

Note: There is not one single right answer as to which feature to add, however you should make sure the feature decreases the Cross Validation RMSE compared to Model 2 (i.e. we want to improve the model, not make it worse!)
This problem will be graded based on your reasoning and explanation of the feature you choose, and then on your implementation of incorporating the feature.

**NOTE** Please don't add additional coding cells below or the Autograder will have issues. You do not need to use all the coding cells provided.

### 0.3.2 Question 5a Answer Cell:

In this cell, explain what feature you chose to add and why. Then give the equation for your new model (use Model 2 from above and then add an additional term).

The custom feature that I am choosing for this model is `Land Square Feet`. The reason why I am choosing this is because properties that have a larger amount of land with them tend to have a higher evaluation. On the contrary, properties that have less land tend to have a lower evaluation. Similar to model 2, I am going to add the log of this parameter to hopefully create a better model over all. The equation for this model will then look like

Model 3

$$\text{Log Sale Price} = \theta_1(\text{Log Building Square Feet}) + \theta_2(\text{Shingle/Asphalt}) + \theta_3(\text{Tar\&Gravel}) + \theta_4(\text{Tile}) \\ + \theta_5(\text{Shake}) + \theta_6(\text{Other}) + \theta_7(\text{Slate}) + \theta_8(\text{Log Land Square Feet})$$

### 0.3.3 Question 5b: Create Model 3

In the cells below fill in the code to create and analyze Model 3 (follow the Modeling steps outlined in Questions 3 and 4).

PLEASE DO NOT ADD ANY ADDITIONAL CELLS IN THIS PROBLEM OR IT MIGHT MAKE THE AUTOGRADER FAIL

In [41]:
```
# Modeling Step 1:  Process the Data

# Hint: You can either use your implementation of the One Hot Encoding Function from Project P

from feature_func import *


def process_data_m3(data):
    # You should start by only keeping values with Pure Market Filter = 1
    data = data[data["Pure Market Filter"] == 1]
    data = ohe_roof_material(data)
    data["Log Sale Price"] = np.log(data["Sale Price"])
    data["Log Building Square Feet"] = np.log(data["Building Square Feet"])
    data["Log Land Square Feet"] = np.log(data["Land Square Feet"])
    columns = ["Log Sale Price", "Log Building Square Feet", "Log Land Square Feet"]
    columns.extend([col for col in data if col.startswith('Roof Material_')])
    processed_data = data[columns]
    return processed_data

# Process the data for Model 3
processed_train_m3 = process_data_m3(train)

processed_val_m3 = process_data_m3(valid)

# Create X (Dataframe) and Y (series) to use to train the model
X_train_m3 = processed_train_m3.drop(columns = "Log Sale Price")
Y_train_m3 = processed_train_m3["Log Sale Price"]

X_valid_m3 = processed_val_m3.drop(columns = "Log Sale Price")
Y_valid_m3 = processed_val_m3["Log Sale Price"]


# Take a look at the result
display(X_train_m3.head())
display(Y_train_m3.head())

display(X_valid_m3.head())
display(Y_valid_m3.head())
```

```
        Log Building Square Feet  Log Land Square Feet  Roof Material_1.0  \
130829                  7.870166              9.172846                1.0
193890                  7.002156              8.338784                1.0
```

```
30507                   6.851185             8.799662                1.0
91308                   7.228388             7.721349                1.0
131132                  7.990915             9.257224                1.0

        Roof Material_2.0  Roof Material_3.0  Roof Material_4.0  \
130829                0.0                0.0                0.0
193890                0.0                0.0                0.0
30507                 0.0                0.0                0.0
91308                 0.0                0.0                0.0
131132                0.0                0.0                0.0

        Roof Material_5.0  Roof Material_6.0
130829                0.0                0.0
193890                0.0                0.0
30507                 0.0                0.0
91308                 0.0                0.0
131132                0.0                0.0


130829    12.994530
193890    11.848683
30507     11.813030
91308     13.060488
131132    12.516861
Name: Log Sale Price, dtype: float64


        Log Building Square Feet  Log Land Square Feet  Roof Material_1.0  \
50636                   7.310550              8.047190                1.0
82485                   7.325808              8.509161                1.0
193966                  7.090077              8.921057                1.0
160612                  7.281386              9.462188                1.0
7028                    7.118016              8.332789                1.0

        Roof Material_2.0  Roof Material_3.0  Roof Material_4.0  \
50636                 0.0                0.0                0.0
82485                 0.0                0.0                0.0
193966                0.0                0.0                0.0
160612                0.0                0.0                0.0
7028                  0.0                0.0                0.0

        Roof Material_5.0  Roof Material_6.0
50636                 0.0                0.0
82485                 0.0                0.0
193966                0.0                0.0
160612                0.0                0.0
7028                  0.0                0.0


50636     11.682668
82485     12.820655
193966     9.825526
160612    12.468437
```

```
7028      12.254863
Name: Log Sale Price, dtype: float64
```

In [42]:
```
# Modeling STEP 2:  Create a Multiple Linear Regression Model

# Be sure to set fit_intercept to False, since we are incorporating one-hot-encoded data

linear_model_m3 = lm.LinearRegression(fit_intercept = False)
linear_model_m3.fit(X_train_m3, Y_train_m3)

# your code above this line to create regression model for Model 2

Y_predict_train_m3 = linear_model_m3.predict(X_train_m3)

Y_predict_valid_m3 = linear_model_m3.predict(X_valid_m3)
```

In [43]:
```
# MODELING STEP 3:  Evaluate the RMSE for your model

# Training and test errors for the model (in its units of Log Sale Price)

training_error_log[2] = rmse(Y_predict_train_m3, Y_train_m3)
validation_error_log[2]= rmse(Y_predict_valid_m3, Y_valid_m3)

y_train_pred_exp = np.exp(Y_predict_train_m3)
y_train_exp = np.exp(Y_train_m3)
y_val_pred_exp = np.exp(Y_predict_valid_m3)
y_val_exp = np.exp(Y_valid_m3)

# Training and test errors for the model (in its original values before the log transform)
training_error[2] = rmse(y_train_pred_exp, y_train_exp)
validation_error[2] = rmse(y_val_pred_exp, y_val_exp)


print("3rd Model\nTraining RMSE (log): {}\nValidation RMSE (log): {}\n".format(training_error_l
print("3rd Model \nTraining RMSE: {}\nValidation RMSE: {}\n".format(training_error[2], validati
```

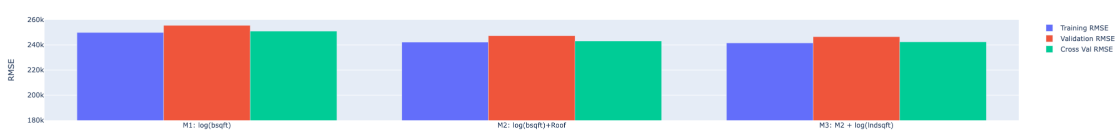
```
3rd Model
Training RMSE (log): 0.7483197519746363
Validation RMSE (log): 0.7479270898764239

3rd Model
Training RMSE: 241605.59600453379
Validation RMSE: 246604.11319280296
```

In [44]:
```
# MODELING STEP 4:  Conduct 5-fold cross validation for model and output RMSE

linear_model_m3_cv = lm.LinearRegression(fit_intercept = False)
```

```
processed_full_m3 = process_data_m3(training_val_data)

X_full_m3 = processed_full_m3.drop(columns = "Log Sale Price")
Y_full_m3 = processed_full_m3["Log Sale Price"]

# DO NOT CHANGE THIS LINE - it ensures reproducibility
np.random.seed(1330)

# your code above this line to use 5-fold cross-validation and output RMSE (in units of dollar

cv_error[2] = cross_validate_rmse(linear_model_m3_cv, X_full_m3, Y_full_m3)

print("3rd Model Cross Validation RMSE: {}".format(cv_error[2]))
```

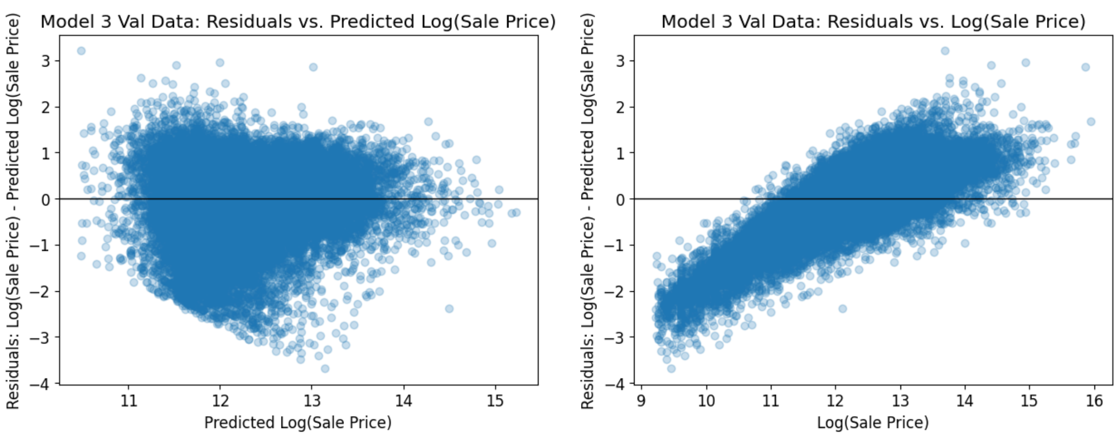
```
3rd Model Cross Validation RMSE: 242424.8195489874
```

In [45]:
```
# MODELING STEP 5:  Add a name for your 3rd model describing the features and run this cell to

model_names[2] = "M3: M2 + log(lndsqft)"


fig = go.Figure([
go.Bar(x = model_names, y = training_error, name="Training RMSE"),
go.Bar(x = model_names, y = validation_error, name="Validation RMSE"),
go.Bar(x = model_names, y = cv_error, name="Cross Val RMSE")
])

fig.update_yaxes(range=[180000,260000], title="RMSE")

fig
```




In [46]:
```
# MODELING STEP 5 cont'd:  Plot 2 side-by-side residual plots (similar to Question 3, for vali

fig, ax = plt.subplots(1,2, figsize=(15, 5))
```

```python
Y_valid_m3_array = Y_valid_m3.to_numpy(dtype = object)

x_plt1 = Y_predict_valid_m3
y_plt1 = Y_valid_m3_array - Y_predict_valid_m3

x_plt2 = Y_valid_m3_array
y_plt2 = y_plt1


ax[0].scatter(x_plt1, y_plt1, alpha=.25)
ax[0].axhline(0, c='black', linewidth=1)
ax[0].set_xlabel(r'Predicted Log(Sale Price)')
ax[0].set_ylabel(r'Residuals: Log(Sale Price) - Predicted Log(Sale Price)');
ax[0].set_title("Model 3 Val Data: Residuals vs. Predicted Log(Sale Price)");

ax[1].scatter(x_plt2, y_plt2, alpha=.25)
ax[1].axhline(0, c='black', linewidth=1)
ax[1].set_xlabel(r'Log(Sale Price)')
ax[1].set_ylabel(r'Residuals: Log(Sale Price) - Predicted Log(Sale Price)');
ax[1].set_title("Model 3 Val Data: Residuals vs. Log(Sale Price)");
```



### 0.3.4 Question 5c

i). Comment on your RMSE and residual plots from Model 3 compared to the first 2 models.

ii). Are the residuals of your model still showing a trend that overestimates lower priced houses and underestimates higher priced houses? If so, how could you try to address this in the next round of modeling?

iii). If you had more time to improve your model, what would your next steps be?

**Part i** The RMSE's of these models are as follows:

| Model Number | Training RMSE | Validation RMSE |
|---|---|---|
| Model 1 | 249,895.91 | 255,533.84 |
| Model 2 | 242,236.37 | 247,381.61 |
| Model 3 | 241,605.60 | 246,604.11 |

We can see that the RMSE for my model is slightly better than that of model 2's for both the Training RMSE and the Validation RMSE. The residual plots for both the predicted log sale price and the valid log sale price are nearly identical to that of the previous two models. Indicating that my model is not much better than the previous model's.

**Part ii** The residuals that are found in the prediction plot are showing a fair model in terms of tax systems. However, the plot of the actual log(Sale Price) are still indicating a regressive tax system. There are a number of ways that we could address this issue. Here are some possible solutions:

- We could add more features to try to more accurately predict the sale price of a property so that the residuals decrease in magnitude.
- We could add a weighted least squares regression, where the lower-priced properties are given more weight to try to counteract the regressive nature.

The easiest change we could make would be adding more features to the model.

**Part iii** If I had more time, I would implement more features into my model. I believe this would decrease the RMSE significantly and produce a more accurate model because it is contingent upon more predictors to accurately place a price on a property.

## 0.4 Question 6: Evaluating the Model in Context

---

## 0.5 Question 6a

When evaluating your model, we used RMSE. In the context of estimating the value of houses, what does the residual mean for an individual homeowner? How does it affect them in terms of property taxes? Discuss the cases where residual is positive and negative separately.

The residual for the homeowner is essentially the difference between the predicted Sale Price and the actual actual sale price. The RMSE is evaluating the difference of sale prices that are transformed with a logarithm, namely

$$\text{Actual Log(Sale Price)} - \text{Predicted Log(Sale Price)}.$$

When a residual is negative, the model is predicting a sale price that is higher than the actual sale price of the property. This disproportionately affects those who have a lower valued property because they will in turn pay more in taxes (in terms of total volume) than those with the opposite residual.

On the contrary, when a residual is positive the model is predicting a sale price that is lower than that of the actual sale price of the property. This in turn causes those who have higher valued property to pay less taxes (in volume) than those who have a lower valued property.

## 0.6 Question 6b

Reflecting back on your exploration in Questions 5 and 6a, in your own words, what makes a model's predictions of property values for tax assessment purposes "fair"?

This question is open-ended and part of your answer may depend upon your specific model; we are looking for thoughtfulness and engagement with the material, not correctness.

**Hint:** Some guiding questions to reflect on as you answer the question above: What is the relationship between RMSE, accuracy, and fairness as you have defined it? Is a model with a low RMSE necessarily accurate? Is a model with a low RMSE necessarily "fair"? Is there any difference between your answers to the previous two questions? And if so, why?

In my opinion, a model's predictions of property values that would be considered fair is a model that is as accurate as possible. Consequently this would mean a model with a lower RMSE would mean that it is more fair than one that had a higher RMSE. If we are seeking to create a system that is fair across the board, then accuracy should be of our highest concerns. It is not fair to undervalue those who have a lower valued property and then overvalue those who have a higher valued property, because we are essentially instilling a system in which that has a bias towards those who have a higher valued property.

I firmly believe if we are seeking to be fair, we should aim to create a system where there are as little biases as possible. Those who are more fortunate than others should not be penalized than those who are less fortunate, in my personal opinion.

To create a model that is fair for tax assessment purposes, we should seek to create a model that is as accurate as possible. Decreasing residuals and the RMSE of these models is the numerical way of achieving this.

## 0.7 Extra Credit Step 1: Creating Your Model

Complete the modeling steps (you can skip the cross validation step to save memory) in the cells below.

DO NOT ADD ANY EXTRA CELLS BELOW (for this part of the problem)

In [50]:
```
# Modeling Step 1:  Process the Data

# Hint: You can either use your implementation of the One Hot Encoding Function from Project P
#from feature_func import *

# Optional:  Define any helper functions you need for one-hot encoding above this line

def add_total_rooms(data):
    """
    Input: data (data frame): a data frame containing at least the Description column.
    """
    rooms_regex = r'(\d+) rooms,'
    rooms = data['Description'].str.extract(rooms_regex).astype(int)
    data['Rooms'] = rooms
    return data

def polynomial_feature(data, feature, power):
    data[f"{feature} Polynomial"] = np.power(data[feature], power)
    return data

def process_data_ec(data, is_test_set=False):
    # Please include all of your feature engineering processes for both
    # the training/validation as well as the test data inside this function.
    data = ohe_roof_material(data)
    data = add_total_rooms(data)
    data = polynomial_feature(data, "Town Code", 2)
    data = polynomial_feature(data, "Basement", 3)
    data = polynomial_feature(data, "Central Air", 2)
    data = polynomial_feature(data, "Fireplaces", 3)
    data = polynomial_feature(data, "Cathedral Ceiling", 3)
    data = polynomial_feature(data, "Multi Code", 2)
    data = polynomial_feature(data, "Census Tract", 2)
    data["Log Building Square Feet"] = np.log(data["Building Square Feet"])
    data["Log Estimate (Building)"] = np.log(data["Estimate (Building)"] != float('inf'))
    data["Log Estimate (Land)"] = np.log(data["Estimate (Land)"] != float('inf'))
    columns = [
        "Log Building Square Feet",
        "Log Estimate (Building)",
        "Log Estimate (Land)",
        "Land Square Feet",
        "Town Code",
        "Town Code Polynomial",
        "Basement",
        "Basement Polynomial",
        "Basement Finish",
```

```
        "Central Air",
        "Central Air Polynomial",
        "Fireplaces",
        "Fireplaces Polynomial",
        "Cathedral Ceiling",
        "Cathedral Ceiling Polynomial",
        "Site Desirability",
        "Other Improvements",
        "Multi Code",
        "Multi Code Polynomial",
        "Census Tract",
        "Census Tract Polynomial",
        "Town and Neighborhood",
        "Rooms",
        "Lot Size"
    ]
    columns.extend([col for col in data if col.startswith('Roof Material_')])
    # Can include feature engineering processes for both the training/validation
    # and the test data above this line
    # Whenever you access 'Log Sale Price' or 'Sale Price', make sure to use the
    # condition is_test_set like this:
    if not is_test_set:
        # Processing for the training/validation set (i.e. not the test set)
        # CAN involve references to sale price!
        # CAN involve filtering certain rows or removing outliers
        data = data[data["Pure Market Filter"]==1]
        data = data[data["Number of Commercial Units"]==0]
        data['Log Sale Price'] = np.log(data['Sale Price'])
        columns.append("Log Sale Price")
        # Include the rest of your feature engineering processes for the
        # training/validation set above this line
    else:
        # Processing for the test set
        # CANNOT involve references to sale price!
        # CANNOT involve removing any rows
        assert test_data.shape[0] == 55311
    return data[columns]

# Process the data
processed_train_ec = process_data_ec(train, is_test_set=False)
processed_val_ec = process_data_ec(valid, is_test_set=False)

X_train_ec = processed_train_ec.drop(columns = "Log Sale Price")
Y_train_ec = processed_train_ec["Log Sale Price"]

X_valid_ec = processed_val_ec.drop(columns = "Log Sale Price")
Y_valid_ec = processed_val_ec["Log Sale Price"]

# Take a look at the result
display(X_train_ec.head())
display(Y_train_ec.head())

display(X_valid_m3.head())
display(Y_valid_m3.head())
```

```
        Log Building Square Feet  Log Estimate (Building)  \
130829                  7.870166                      0.0
193890                  7.002156                      0.0
30507                   6.851185                      0.0
91308                   7.228388                      0.0
131132                  7.990915                      0.0

        Log Estimate (Land)  Land Square Feet  Town Code  \
130829                  0.0            9632.0         38
193890                  0.0            4183.0         72
30507                   0.0            6632.0         31
91308                   0.0            2256.0         77
131132                  0.0           10480.0         13

        Town Code Polynomial  Basement  Basement Polynomial  Basement Finish  \
130829                  1444       1.0                  1.0              3.0
193890                  5184       2.0                  8.0              3.0
30507                    961       1.0                  1.0              1.0
91308                   5929       1.0                  1.0              3.0
131132                   169       3.0                 27.0              3.0

        Central Air  …  Census Tract Polynomial  Town and Neighborhood  \
130829          1.0  …             6.441748e+11                   3845
193890          0.0  …             2.407865e+11                  72330
30507           1.0  …             6.663457e+11                   3141
91308           1.0  …             5.904900e+10                  77120
131132          1.0  …             6.811234e+11                  13182

        Rooms  Lot Size  Roof Material_1.0  Roof Material_2.0  \
130829      8    9632.0                1.0                0.0
193890      4    4183.0                1.0                0.0
30507       4    6632.0                1.0                0.0
91308       5    2256.0                1.0                0.0
131132      8   10480.0                1.0                0.0

        Roof Material_3.0  Roof Material_4.0  Roof Material_5.0  \
130829                0.0                0.0                0.0
193890                0.0                0.0                0.0
30507                 0.0                0.0                0.0
91308                 0.0                0.0                0.0
131132                0.0                0.0                0.0

        Roof Material_6.0
130829                0.0
193890                0.0
30507                 0.0
91308                 0.0
131132                0.0

[5 rows x 30 columns]


130829    12.994530
193890    11.848683
```

```
30507     11.813030
91308     13.060488
131132    12.516861
Name: Log Sale Price, dtype: float64


        Log Building Square Feet  Log Land Square Feet  Roof Material_1.0  \
50636                   7.310550              8.047190                1.0
82485                   7.325808              8.509161                1.0
193966                  7.090077              8.921057                1.0
160612                  7.281386              9.462188                1.0
7028                    7.118016              8.332789                1.0

        Roof Material_2.0  Roof Material_3.0  Roof Material_4.0  \
50636                 0.0                0.0                0.0
82485                 0.0                0.0                0.0
193966                0.0                0.0                0.0
160612                0.0                0.0                0.0
7028                  0.0                0.0                0.0

        Roof Material_5.0  Roof Material_6.0
50636                 0.0                0.0
82485                 0.0                0.0
193966                0.0                0.0
160612                0.0                0.0
7028                  0.0                0.0


50636     11.682668
82485     12.820655
193966     9.825526
160612    12.468437
7028      12.254863
Name: Log Sale Price, dtype: float64
```

In [51]:
```python
# Modeling STEP 2:  Create a Multiple Linear Regression Model

# If you are are incorporating one-hot-encoded data, set the fit_intercept to False

linear_model_ec = lm.LinearRegression(fit_intercept = False)
linear_model_ec.fit(X_train_ec, Y_train_ec)

# your code above this line to create regression model for Model 2

Y_predict_train_ec = linear_model_ec.predict(X_train_ec)

Y_predict_valid_ec = linear_model_ec.predict(X_valid_ec)
```

In [52]:
```python
# MODELING STEP 3:  Evaluate the RMSE for your model

training_error_log = rmse(Y_predict_train_ec, Y_train_ec)
```

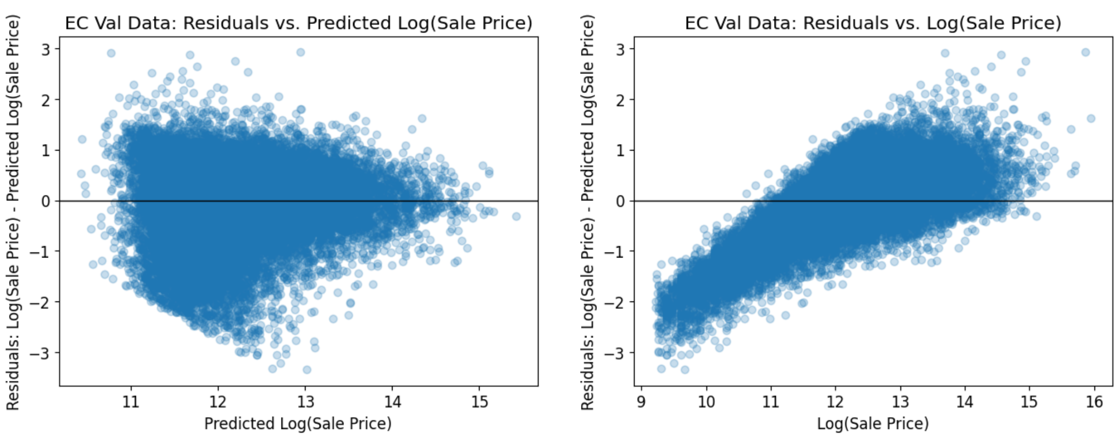
```
validation_error_log = rmse(Y_predict_valid_ec, Y_valid_ec)

y_train_pred_exp = np.exp(Y_predict_train_ec)
y_train_exp = np.exp(Y_train_ec)
y_val_pred_exp = np.exp(Y_predict_valid_ec)
y_val_exp = np.exp(Y_valid_ec)

# Training and test errors for the model (in its original values before the log transform)
training_error_ec = rmse(y_train_pred_exp, y_train_exp)
validation_error_ec = rmse(y_val_pred_exp, y_val_exp)

print("Extra Credit Model\nTraining RMSE (log): {}\nValidation RMSE (log): {}\n".format(trainin
print("Extra Credit \nTraining RMSE: {}\nValidation RMSE: {}\n".format(training_error_ec, vali
```

```
Extra Credit Model
Training RMSE (log): 0.6875751101720305
Validation RMSE (log): 0.6875361628489552

Extra Credit
Training RMSE: 212211.2974452848
Validation RMSE: 216450.72735293876
```

In [53]:
```
# Optional: Run this cell to visualize

fig = go.Figure([
go.Bar(x = ["Extra Credit Model"], y = [training_error_ec], name="Training RMSE"),
go.Bar(x = ["Extra Credit Model"], y = [validation_error_ec], name="Validation RMSE"),

])


fig
fig.update_yaxes(range=[50000,250000], title="RMSE")
```


In [54]:
```
# MODELING STEP 5: Plot 2 side-by-side residual plots for validation data
```

```python
Y_valid_ec_array = Y_valid_ec.to_numpy(dtype = object)

fig, ax = plt.subplots(1,2, figsize=(15, 5))


x_plt1 = Y_predict_valid_ec
y_plt1 = Y_valid_ec_array - Y_predict_valid_ec

x_plt2 = Y_valid_ec_array
y_plt2 = y_plt1


ax[0].scatter(x_plt1, y_plt1, alpha=.25)
ax[0].axhline(0, c='black', linewidth=1)
ax[0].set_xlabel(r'Predicted Log(Sale Price)')
ax[0].set_ylabel(r'Residuals: Log(Sale Price) - Predicted Log(Sale Price)');
ax[0].set_title("EC Val Data: Residuals vs. Predicted Log(Sale Price)");

ax[1].scatter(x_plt2, y_plt2, alpha=.25)
ax[1].axhline(0, c='black', linewidth=1)
ax[1].set_xlabel(r'Log(Sale Price)')
ax[1].set_ylabel(r'Residuals: Log(Sale Price) - Predicted Log(Sale Price)');
ax[1].set_title("EC Val Data: Residuals vs. Log(Sale Price)");
```



## 0.8 Extra Credit Step 2: Explanation (Required for points on model above):

Explain what you did to create your model. What versions did you try? What worked and what didn't?

Comment on the RMSE and residual plots from your model. Are the residuals of your model still showing a trend that overestimates lower priced houses and underestimates higher priced houses?

For starters, I added as many features to my model as possible. I then went and transformed some of these features by turning them into polynomial features or by taking the logarithm of them. In the training aspect of this model I filtered out as many outliers as I possibly could. Coupling this together, I was able to get the RMSE down to around $\approx 215k$. In turn, the residuals for these outliers decreased in both directions and it appears that my model is doing a pretty good job at predicting house prices.