0.0.1 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.**

- Property investment/investment companies, who would want the price to be high if selling, and low if buying. They’ll also probably be interested to know if the area is developing.
- House buyers, who would want the price to be low and within their budget.
- The government, who might have an interest in seeing the housing prices high to increase property tax revenue.

0.0.2 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.

All of the scenarios are unfair.

A, is unfair because it means the homeowner is paying more in property tax than they should be. Also would give the homeowner false hope that their home has appreciated more than it actually did

B, because if my home was assessed at a lower valuation than it should be, it means im not getting as big of a return on my investment which means I lose money. Selling at a loss (compared to market value) is like doing charity. You would pay less in property tax though.

C, is very unfair because it would mean that the people who can afford expensive properties are getting a break, while the people who can't afford expensive properties are getting screwed. Lower income properties will be priced higher than they should be making them inaccessible.

D, is unfair because it does not accurately reflect the market value of the proprty. But because majority of the people are not rich and don't buy expensive properties, it shifts the burden of property tax to owner's of expensive properties who are more likely to be able to afford them.

0.0.3 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 central problems with the earlier property tax system in Cook County was that lower priced homes were being overvalued while higher priced homes were being undervalued. The primary cause to these problems were not to do with the models the county assessor's office was using but the appeal system that allowed wealthier homeowners with better legal representation to appeal their property assessments. Having more time, money and accessibility to better legal representation allowed wealthier homeowners challenge their property assesments more successfully. This system created a system which favored those with financial means and better legal representation.

0.1 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.

0.1.1 Question 2a i, ii, iii answer cell:**

2ai) There are no 0 or null values in the dataset but there is an unproportional amount of houses with the price of 1. This might be a placeholder value for missing data. Searching online, it seems that $1 is a placeholder for non-market transactions such as transfers between family members (https://www.quora.com/What-does-it-mean-when-a-property-is-sold-for-1). I am not sure how accurate the source is but that's the most I can gather from this. I believe it would be reasonable to remove any Sale Prices that are less than $100 as they are likely to be a placeholder or unrepresentative of the actual value of the property (https://prodassets.cookcountyassessor.com/s3fs-public/reports/PCS_Cook-County_CSRS_v1.pdf)

2aii) Sale Price does have unusually large outlier values which are most likely just very expensive properties. To determine the cutoff, I propose using a cut off similar to the one I suggested in part 1 of this project. That is, we can overcome this issue is by only including houses within and slightly beyond the whiskers of the box-plot as that is where we can expect most of our house prices to be within. That is we should limit our Sale Price to only contain prices within $[4.52 \cdot 10^4 - (2 \cdot (3.12 \cdot 10^5 - 4.52 \cdot 10^4)), 3.12 \cdot 10^5 + (2 \cdot (3.12 \cdot 10^5 - 4.52 \cdot 10^4))]$ Because the calculation for the lower end will be negative, we can just set it to 0.

2aiii) Sale Price does have unusually small outlier values as I explained in 2ai. I propose using a cutoff of $100 as that is the cut off used by the cook county assessor's office (according to that pdf i found) for their own analysis and studies.

```
In [280]: print(training_val_data['Sale Price'].describe())

          # hmm the min of 1 is kind of suspicious but I'm going to check the number of null values fir
          print("Number of null values: ", training_val_data['Sale Price'].isnull().sum())

          # Is the null values replaced with 1 in this dataset?
          print("Number of sales with price 1: ", (training_val_data['Sale Price'] == 1).sum())

          # Checking if it is actually normal for the prices of houses to be 1
          print("Number of sales with price between 1 (exclusive) and 1000:",
                ((training_val_data['Sale Price'] > 1) & (training_val_data['Sale Price'] <= 100)).sum(

          print("Ratio of sales above 75% + 2 * IQR: ",
                ((training_val_data['Sale Price'] > training_val_data['Sale Price'].quantile(0.75)
                  + 2 * (training_val_data['Sale Price'].quantile(0.75)
                         - training_val_data['Sale Price'].quantile(0.25))).sum())
                         /training_val_data.shape[0])

          # hmm 6% of data being above the 75% + 1.5 * IQR is a bit high so i'll try 2 * IQR
          print("75% + 2 * IQR: ",
                training_val_data['Sale Price'].quantile(0.75)
                + 2 * (training_val_data['Sale Price'].quantile(0.75)
                         - training_val_data['Sale Price'].quantile(0.25)))

          # your code exploring Sale Price above this line
```

```
count    2.047920e+05
mean     2.451646e+05
std      3.628694e+05
min      1.000000e+00
25%      4.520000e+04
50%      1.750000e+05
75%      3.120000e+05
max      7.100000e+07
Name: Sale Price, dtype: float64
Number of null values:  0
Number of sales with price 1:  35546
Number of sales with price between 1 (exclusive) and 1000: 300
Ratio of sales above 75% + 2 * IQR:  0.04169108168287824
75% + 2 * IQR:  845600.0
```

0.1.2 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.

0.1.3 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).

Based on the data, it appears that Estimate (Building) would be a good feature to add to the model as it is quite strongly correlated with the sale price of a propoerty. Estimate (Building) represent's the building's estimated market value for the previous tax year. This is generally a good indicator of the market value of a property. However, this means that the model will be reliant on buildings having a prior tax year market value and the precision and fairness of these estimates.

Logically, though, prior market valuation will provide a good idea on the sale price of a property. This connection outweighs the downsides of using this feature.

Our new model will be:

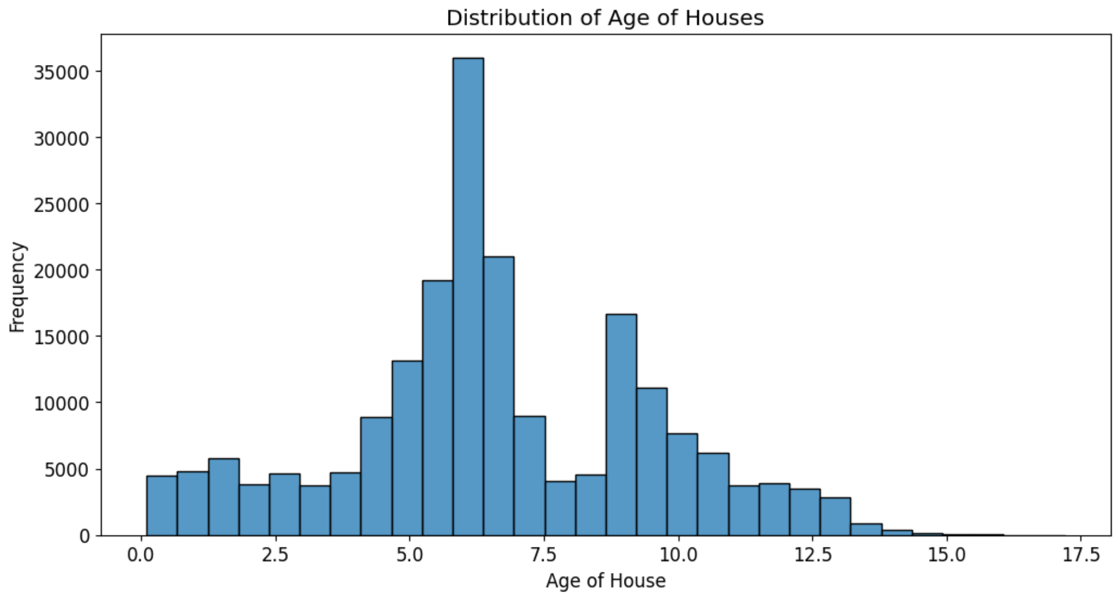
$$\text{Log Sale Price} = \theta_1(\text{Log Building Square Feet}) + \theta_2(\text{Shingle/Asphalt}) + \theta_3(\text{Tar\&Gravel}) + \theta_4(\text{Tile}) + \theta_5(\text{Shake}) + \theta_6(\text{Other}) + \theta_7(\text{S}$$

```
In [311]: training_val_data['Road Proximity'].value_counts()

          # Show work in this cell exploring data to determine which feature to add


Out[311]: 0.0     190453
```

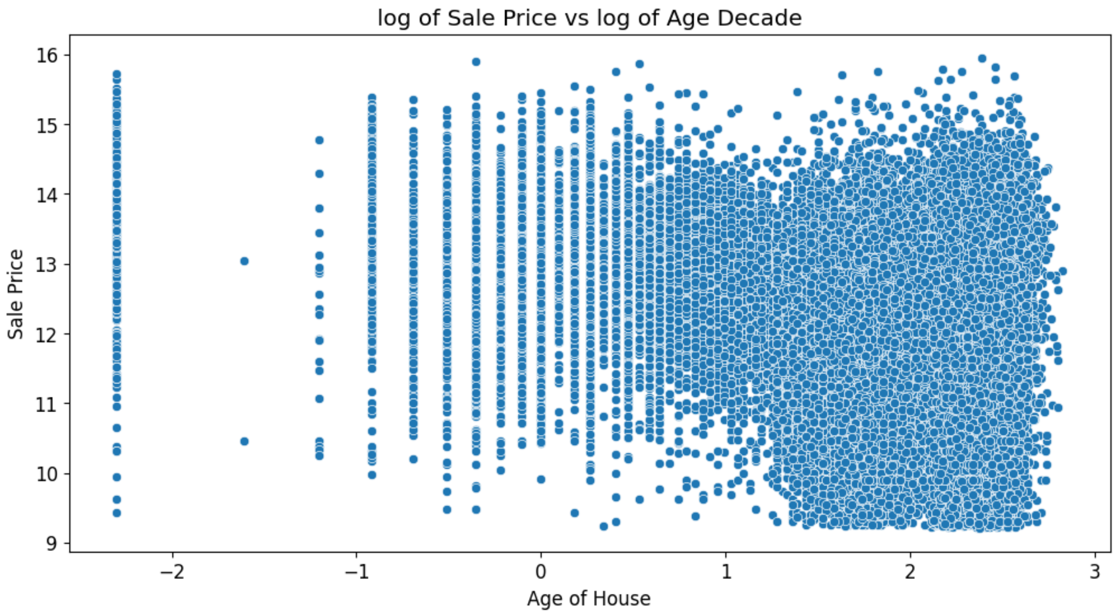
```
1.0     14339
Name: Road Proximity, dtype: int64
```

```
In [312]: display(training_val_data['Age Decade'].describe())

          # visualising the distribution of the age of the houses
          plt.figure(figsize=(12, 6))
          sns.histplot(training_val_data['Age Decade'], bins=30)
          plt.title("Distribution of Age of Houses")
          plt.xlabel("Age of House")
          plt.ylabel("Frequency")
          plt.show()

          # finding the correlation between log Age Decade and log Sale Price
          log_age_decade = np.log(training_val_data['Age Decade'])
          log_sale_price = np.log(training_val_data['Sale Price'])
          correlation = log_age_decade.corr(log_sale_price)
          print(f"Correlation between log of Age Decade and log of Sale Price: {correlation}")


          plt.figure(figsize=(12, 6))
          sns.scatterplot(x=np.log(training_val_data[training_val_data['Pure Market Filter'] == 1]['Age
          plt.title("log of Sale Price vs log of Age Decade")
          plt.xlabel("Age of House")
          plt.ylabel("Sale Price")
          plt.show()

          # Optional code cell for additional work exploring data/ explaining which feature you chose.
```

```
count    204792.000000
mean          6.598121
std           2.900149
min           0.100000
25%           5.100000
50%           6.200000
75%           8.900000
max          17.200000
Name: Age Decade, dtype: float64
```

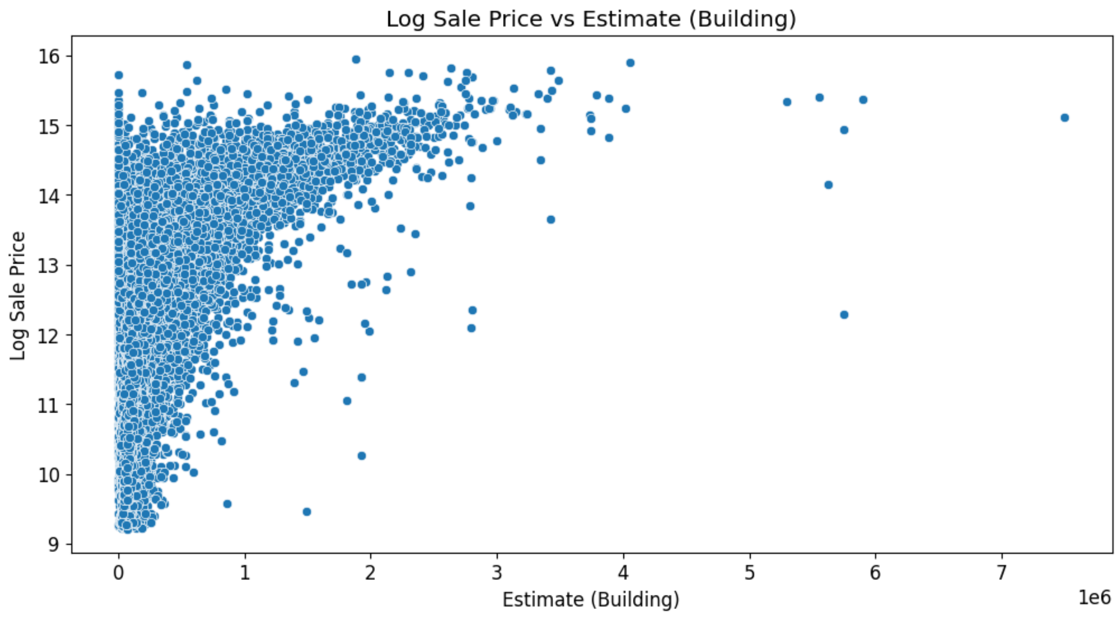


```
Correlation between log of Age Decade and log of Sale Price: -0.12525287157471637
```



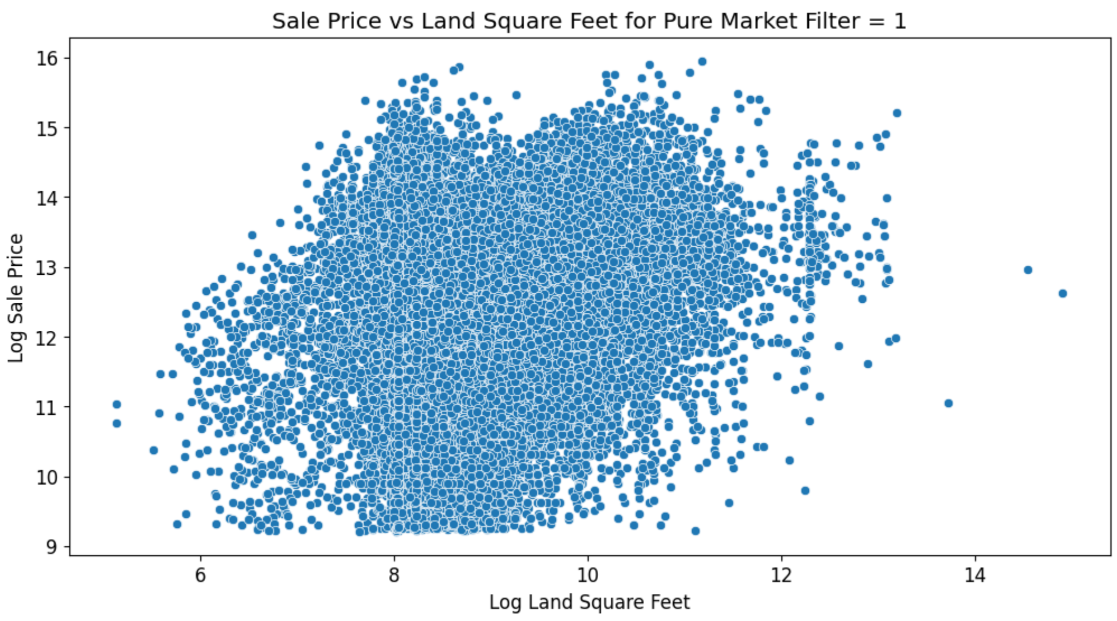
```
In [313]: display(sum(training_val_data[training_val_data['Pure Market Filter'] == 1]['Estimate (Buildi

          plt.figure(figsize=(12, 6))
          sns.scatterplot(x=training_val_data[training_val_data['Pure Market Filter'] == 1]['Estimate (
          plt.title("Log Sale Price vs Estimate (Building)")
          plt.xlabel("Estimate (Building)")
          plt.ylabel("Log Sale Price")
          plt.show()


          print("Correlation between Estimate (Building) and Sale Price: ", training_val_data['Estimate
```

```
1235
```



```
Correlation between Estimate (Building) and Sale Price:  0.6092859158684686
```

```
In [314]: # scatter plot land square feet vs sale price

          plt.figure(figsize=(12, 6))
          filtered_data = training_val_data[training_val_data['Pure Market Filter'] == 1]
          sns.scatterplot(x=np.log(filtered_data['Land Square Feet']), y=np.log(filtered_data['Sale Pri
          plt.title("Sale Price vs Land Square Feet for Pure Market Filter = 1")
          plt.xlabel("Log Land Square Feet")
          plt.ylabel("Log Sale Price")
          plt.show()

          # finding correlation

          log_land_sqft = np.log(filtered_data['Land Square Feet'])
          correlation = log_land_sqft.corr(log_sale_price)
          print(f"Correlation between log of Land Square Feet and log of Sale Price: {correlation}")
```



```
Correlation between log of Land Square Feet and log of Sale Price: 0.2578511813965192
```

```
In [315]: # Modeling Step 1:  Process the Data

          # Hint: You can either use your implementation of the One Hot Encoding Function
          #from Project Part 1, or use the staff's implementation

          from feature_func import *


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


          def process_data_m3(data):

              # You should start by only keeping values with Pure Market Filter = 1

              # Remove Non-Market Sales
              data = data[data["Pure Market Filter"]==1]

              # Create Log Sale Price column
              data["Log Sale Price"] = np.log(data["Sale Price"])

              # Create Log Building Square Feet column
              data["Log Building Square Feet"] = np.log(data["Building Square Feet"])

              # Create Log Estimate (Building) column
              data["Log Estimate (Building)"] = np.log(data["Estimate (Building)"] + 1)

              # Change Roof Material to names
              data = substitute_roof_material(data)

              # one-hot encode the roof material
              data = ohe_roof_material(data)

              # select only relevant columns
              relevant_columns = ['Log Estimate (Building)', 'Log Building Square Feet', 'Log Sale Pric
              data = data[relevant_columns]



              return data


          # Use the same original train and valid datasets from 3a
          #(otherwise the validation errors aren't comparable),
          # Don't resplit the 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 Estimate (Building)  Log Building Square Feet  \
130829                13.019934                  7.870166
193890                10.969766                  7.002156
30507                 11.569599                  6.851185
91308                 12.839685                  7.228388
131132                12.357553                  7.990915

        Roof Material_Other  Roof Material_Shake  \
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

        Roof Material_Shingle/Asphalt  Roof Material_Slate  \
130829                            1.0                  0.0
193890                            1.0                  0.0
30507                             1.0                  0.0
91308                             1.0                  0.0
131132                            1.0                  0.0

        Roof Material_Tar&Gravel  Roof Material_Tile
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 Estimate (Building)  Log Building Square Feet  \
50636                 11.669767                  7.310550
```

```
82485                 12.264676                  7.325808
193966                10.669908                  7.090077
160612                12.154100                  7.281386
7028                  11.355581                  7.118016

        Roof Material_Other  Roof Material_Shake  \
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

        Roof Material_Shingle/Asphalt  Roof Material_Slate  \
50636                             1.0                  0.0
82485                             1.0                  0.0
193966                            1.0                  0.0
160612                            1.0                  0.0
7028                              1.0                  0.0

        Roof Material_Tar&Gravel  Roof Material_Tile
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 [316]: # 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)

          # Fit the model using the processed training data
          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 [317]: # 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)

# Training and test errors for the model (in its original values before the log transform)
training_error[2] = rmse(np.exp(Y_predict_train_m3), np.exp(Y_train_m3))
validation_error[2] = rmse(np.exp(Y_predict_valid_m3), np.exp(Y_valid_m3))


(print("3rd Model\nTraining RMSE (log): {}\nValidation RMSE (log): {}\n"
      .format(training_error_log[2], validation_error_log[2]))
)

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

```
3rd Model
Training RMSE (log): 0.7240559890113686
Validation RMSE (log): 0.7217962345952899

3rd Model
Training RMSE: 231705.33504415533
Validation RMSE: 236566.6507003559
```

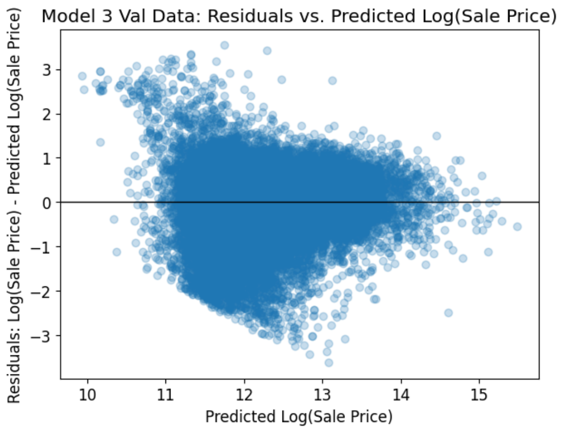
```
In [318]: # MODELING STEP 4:  Conduct 5-fold cross validation for model and output RMSE

          # Create a new model to fit on the whole training_val dataset
          linear_model_m3_cv = lm.LinearRegression(fit_intercept=False)

          # Process the entire training_val dataset using the pipeline
          processed_full_m3 = process_data_m3(training_val_data)

          # Split into X and Y:
          X_full_m3 = processed_full_m3.drop(columns = "Log Sale Price")
          Y_full_m3 = processed_full_m3["Log Sale Price"]

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

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

          cv_error[2] = cross_validate_rmse(linear_model_m3, X_full_m3, Y_full_m3)

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

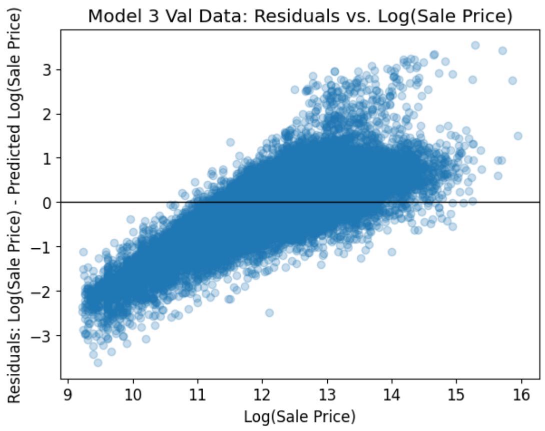
```
3rd Model Cross Validation RMSE: 232477.29879015923
```

```
In [319]: # MODELING STEP 5:  Add a name for your 3rd model describing the features
          #and run this cell to Plot bar graph all 3 models

          model_names[2] = "M3: log(bsqft)+log(est_bldg)+Roof"


          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 [320]: # MODELING STEP 5 cont'd:  Plot 2 side-by-side residual plots
          #(similar to Question 3, for validation data)

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


          x_plt1 = Y_predict_valid_m3
          y_plt1 = Y_valid_m3 - Y_predict_valid_m3

          x_plt2 = Y_valid_m3
          y_plt2 = Y_valid_m3 - Y_predict_valid_m3


          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)")
```

```
Out[320]: Text(0.5, 1.0, 'Model 3 Val Data: Residuals vs. Log(Sale Price)')
```


Model 3 Val Data: Residuals vs. Predicted Log(Sale Price)
Residuals: Log(Sale Price) - Predicted Log(Sale Price)
Predicted Log(Sale Price)


Model 3 Val Data: Residuals vs. Log(Sale Price)
Residuals: Log(Sale Price) - Predicted Log(Sale Price)
Log(Sale Price)

0.1.4 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?

i) The RMSE of Model 3 is significantly lower than both Models 1 and 2. Comparing the residual plot of Model 3 to Models 1 and 2, we can see that it is much more evenly distributed around 0. It more accurately estimates the sale prices of more expensive properties and does not overestimate lower priced houses as much as the previous models.

ii) The residuals still show a trend of overestimating lower priced houses and underestimating higher priced houses. To address this, we could try to add more features to the model to try capture more patterns within the dataset but this might cause the model to be overfitted. We might have to look at using other machine learning models to capture other patterns that a linear regression model might not be able to capture. Qualitative features like Zip Code cannot be used in linear regressioni models but could be used in other models.

iii) If i have more time, maybe it might be possible to get average estimated market value for each zipcode and using that to scale building square feet to increase its importance for wealthier neighbourhoods. Another thing that might be possible is to check the density of a neighbourhood and use that as a feature.

0.2 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.

In the context of estimating house values, the residual represents the difference between the actual sale price of a house and the price predicted by our model. For an individual homeowner, this difference can impact their property taxes in different ways depending on the sign of the residual.

When the residual is positive, it means that our model has underestimated the value of a house. For the homeowner, this could result in lower property taxes since the assessed value used for tax calculation is less than the actual market value. It could also mean that the homeowner not know the full value of their property if they decide to sell.

Conversely, a negative residual indicates that our model has overestimated the value of a house. This could lead to higher property taxes for the homeowner, as the assessed value is higher than it should be. This scenario is bad for homeowners especially since our model tends to overestimate lower cost properties. This means the people who can afford it the least are the ones who are paying the most, proportionally, in property taxes.

0.3 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?

The fairness of a model wouldn't just be based off minimising its RMSE. A fair model should also not make significant amounts of errors and should not be affected by socio-economic status of homeowners and biases of the law.

In this situation we want to make sure that the residual plot is evenly and tightly distributed around 0 as for many, being unfairly taxed can be a huge financial burden.

0.4 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 [325]: # Modeling Step 1:  Process the Data




          # Hint: You can either use your implementation of the One Hot Encoding Function from
          #Project Part 1, or use the staff's implementation
          from feature_func import *
          from sklearn.preprocessing import StandardScaler
          from sklearn.linear_model import Ridge
          from sklearn.linear_model import Lasso


          # Optional:  Define any helper functions you need (for example, for one-hot
          #encoding, etc) above this line


          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.

              # 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

                  # Q1 = data["Sale Price"].quantile(0.25)
                  # Q3 = data["Sale Price"].quantile(0.75)
                  # IQR = Q3 - Q1
                  # upper_limit = Q3 + 2.5 * IQR
                  data = data[(data["Pure Market Filter"]==1) & (data["Sale Price"] >= 100) ]
                  # Create Log Sale Price column
                  data["Log Sale Price"] = np.log(data["Sale Price"])

                  # # Get density of each zip code
                  # neighborhood_density = data['Neighborhood Code'].value_counts(normalize=True)

                  # # Create density_building_square_feet
                  # density_building_square_feet = data['Building Square Feet'] * data['Neighborhood Co
```

```
    # # Create Estimate Building Density
    # avg_estimate_building = data.groupby('Neighborhood Code')['Estimate (Building)'].me

    # estimate_building_density = density_building_square_feet * data['Neighborhood Code']

    # # Create Log Estimate Building Density
    # data['Log Estimate Building Density'] = np.log(estimate_building_density)

    relevant_columns = [ '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

    # # Get density of each zip code
    # neighborhood_density = data['Neighborhood Code'].value_counts(normalize=True)

    # # Create density_building_square_feet
    # density_building_square_feet = data['Building Square Feet'] * data['Neighborhood Co

    # # Create Estimate Building Density
    # avg_estimate_building = data.groupby('Neighborhood Code')['Estimate (Building)'].me

    # estimate_building_density = density_building_square_feet * data['Neighborhood Code']

    # Create Log Estimate Building Density
    # data['Log Estimate Building Density'] = np.log(estimate_building_density)

    relevant_columns = []

scaler = StandardScaler()
data['Standardized Age'] = scaler.fit_transform(data[['Age']])

# Create Log Building Square Feet column
data["Log Building Square Feet"] = np.log(data["Building Square Feet"] + 1)

# Create Log Estimate (Building) column
data["Log Estimate (Building)"] = np.log(data["Estimate (Building)"] +1)

# Create Log Estimate (Land) column
data["Log Estimate (Land)"] = np.log(data["Estimate (Land)"] + 1)
# Add any remaining processing for both test and training set below
#(hint - easiest to put any calls to your One Hot Encoding function here):

# one hot encoding for property class
property_classes = [200, 202, 203, 204, 205, 206, 207, 208, 209, 278, 234, 295]
for pc in property_classes:
    data[f'Property_Class_{pc}'] = (data['Property Class'] == pc).astype(int)
```

```
        relevant_columns = relevant_columns + ['Log Building Square Feet', 'Log Estimate (Buildin
        relevant_columns += [f'Property_Class_{pc}' for pc in property_classes]


        data = data[relevant_columns]
        return data


    # Use the same original train and valid datasets from 3a (otherwise the
    # validation errors aren't comparable).  Don't resplit the data.

    # Process the data
    processed_train_ec = process_data_ec(train)

    processed_val_ec = process_data_ec(valid)


    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_ec.head())
    display(Y_valid_ec.head())
```

```
        Log Building Square Feet  Log Estimate (Building)  Standardized Age  \
130829                  7.870548                13.019934         -1.308199
193890                  7.003065                10.969766          1.061552
30507                   6.852243                11.569599         -0.158173
91308                   7.229114                12.839685         -1.377898
131132                  7.991254                12.357553         -1.203651

        Log Estimate (Land)  Property_Class_200  Property_Class_202  \
130829            10.619643                   0                   0
193890            10.043293                   0                   0
30507             10.671301                   0                   1
91308             10.940242                   0                   0
131132            11.006440                   0                   0

        Property_Class_203  Property_Class_204  Property_Class_205  \
130829                   0                   0                   0
193890                   1                   0                   0
30507                    0                   0                   0
91308                    0                   0                   0
131132                   0                   0                   0

        Property_Class_206  Property_Class_207  Property_Class_208  \
130829                  0                  0                  0
193890                  0                  0                  0
30507                   0                  0                  0
91308                   0                  1                  0
131132                  0                  0                  0

        Property_Class_209  Property_Class_278  Property_Class_234  \
130829                  0                  1                  0
193890                  0                  0                  0
30507                   0                  0                  0
91308                   0                  0                  0
131132                  0                  1                  0

        Property_Class_295
130829                  0
193890                  0
30507                   0
91308                   0
131132                  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 Estimate (Building)  Standardized Age  \
50636                   7.311218                11.669767          1.842927
82485                   7.326466                12.264676          0.474693
193966                  7.090910                10.669908         -0.156799
160612                  7.282074                12.154100         -0.016468
7028                    7.118826                11.355581          1.000937

        Log Estimate (Land)  Property_Class_200  Property_Class_202  \
50636             10.937579                   0                   0
82485             10.649251                   0                   0
193966             9.932609                   0                   0
160612            10.560774                   0                   0
7028              10.472799                   0                   0

        Property_Class_203  Property_Class_204  Property_Class_205  \
50636                   1                  0                  0
82485                   0                  0                  1
193966                  1                  0                  0
160612                  1                  0                  0
7028                    1                  0                  0

        Property_Class_206  Property_Class_207  Property_Class_208  \
50636                   0                  0                  0
82485                   0                  0                  0
```

```
193966                   0                   0                   0
160612                   0                   0                   0
7028                     0                   0                   0

        Property_Class_209  Property_Class_278  Property_Class_234  \
50636                    0                   0                   0
82485                    0                   0                   0
193966                   0                   0                   0
160612                   0                   0                   0
7028                     0                   0                   0

        Property_Class_295
50636                    0
82485                    0
193966                   0
160612                   0
7028                     0


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

```
In [326]: # 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)
```

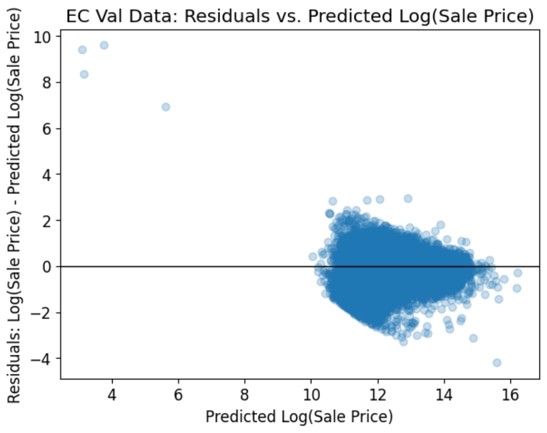
```
In [327]: # MODELING STEP 3:  Evaluate the RMSE for your model


          # Training and test errors for the model
          #(in its original values before the log transform)

          training_error_ec = rmse(np.exp(Y_predict_train_ec), np.exp(Y_train_ec))
          validation_error_ec = rmse(np.exp(Y_predict_valid_ec), np.exp(Y_valid_ec))


          (print("Extra Credit \nTraining RMSE: {}\nValidation RMSE: {}\n"
                 .format(training_error_ec, validation_error_ec))
          )
```

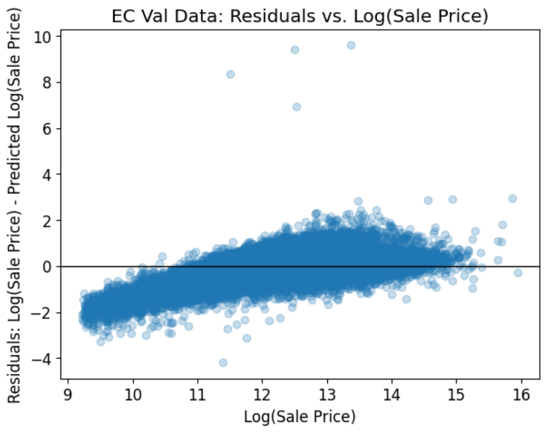
```
Extra Credit
Training RMSE: 207040.79633981557
Validation RMSE: 206919.13392923432
```

```
In [328]: # Optional: Run this cell to visualize

          import plotly.graph_objects as go

          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=[140000,260000], title="RMSE")
          # Feel free to update the range as needed
```

```
In [329]: # MODELING STEP 5: Plot 2 side-by-side residual plots for validation data

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


          x_plt1 = Y_predict_valid_ec
          y_plt1 = Y_valid_ec - Y_predict_valid_ec

          x_plt2 = Y_valid_ec
          y_plt2 = Y_valid_ec - Y_predict_valid_ec


          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)")
```

```
Out[329]: Text(0.5, 1.0, 'EC Val Data: Residuals vs. Log(Sale Price)')
```


EC Val Data: Residuals vs. Predicted Log(Sale Price)
Residuals: Log(Sale Price) - Predicted Log(Sale Price)
Predicted Log(Sale Price)


EC Val Data: Residuals vs. Log(Sale Price)
Residuals: Log(Sale Price) - Predicted Log(Sale Price)
Log(Sale Price)

0.5 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 my model I tried taking the average of the estimated market value for each zipcode and used that to scale the building square feet. This was to increase the importance of square footage in densely populated neighborhoods especially the wealthier ones. But because I did not really get good results with that I decided to ditch it. It gave a promising correlation to sale price but ultimately was not a good predictor.

My model still shows the same trend of overestimating lower priced houses and underestimating higher priced houses. It might be that the data does not include enough metrics that can capture the nuances of property prices such as crime rates of the area, stigma, etc.