0.0.1 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.

1a.

Homeowners, Potential Homeowners(House buyers), IRS, Real Estate Companies

1b.

A and C seems unfair because if A happened, more taxes could be imposed on properties by the unreally higher house price which is property values that they don't actually have. Also if C happened, it would be easier to own houses for wealthy people but those who are in relatively not wealthy groups would pay much higher taxes for their properties. That would exacerbate segregation between them. Also A and C can remove opertunities for whom wanting to move out of not wealthy groups.

0.0.2 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 problem is that property values are assessed wrong because wealthy people's properties were under-assessed but not wealthy people's proerties were over-assessed. The primary causes are as follow.

1. Appeals
   - The appeal system made the problem worse because wealty people can have better and more chances to consult with real estate lawyers, therefore, they could challenge CCAO to correct their property value while others couldn't have enough opportunities.
2. Discrimination in the federal policy
   - The discriminating policy made it difficult or impossible to get a federally backed mortgage to buy a house in specidif neighborhoods coded as "risky"
   - The methods of rating systems for the appraisal of property values has race as a factor of valuation.

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 2abc answer cell:** *Put your answers in this cell...*

2ai) No

2aii) Yes, The maximum price is $ 71,000,000. $7,000,000 could be a good enough cutoff because $7 million is already big enough to cover normal house price

2aiii) Yes, The minimum price is $ 1. $5,000 could be a cutoff because normally house price is higher than that.

```
In [9]: training_val_data['Sale Price'].isna().sum()
        training_val_data['Sale Price'].describe()
        # your code exploring Sale Price above this line
```

```
Out[9]: 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
```

```
In [10]: training_val_data['Sale Price'].info()
         training_val_data['Sale Price'].describe()
         # optional extra cell for exploring code
```

```
<class 'pandas.core.series.Series'>
Index: 204792 entries, 0 to 204791
Series name: Sale Price
Non-Null Count   Dtype
--------------   -----
204792 non-null  int64
dtypes: int64(1)
memory usage: 3.1 MB
```

```
Out[10]: 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
```

0.1.2 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).

I chose "Basement" as a feature for Model3 because Basement type effects house price a lot.

```
In [47]: def substitute_Basement(data):
             """
             Input:
               data (data frame): a data frame containing a 'Roof Material' column.  Its values
                                  should be limited to those found in the codebook
             Output:
               data frame identical to the input except with a refactored 'Roof Material' column
             """
             replacements = {
                 'Basement': {
                     1: 'Full',
                     2: 'Slab',
                     3: 'Partial',
                     4: 'Crawl'

                 }
             }

             data = data.replace(replacements)
             return data

         def ohe_Basement(data):
             """
             One-hot-encodes roof material.  New columns are of the form x0_MATERIAL.
             """
             ...
             # BEGIN SOLUTION NO PROMPT
             oh_enc = OneHotEncoder()
             oh_enc.fit(data[['Basement']])
             dummies = pd.DataFrame(oh_enc.transform(data[['Basement']]).todense(),
                                    columns=oh_enc.get_feature_names_out(),
                                    index = data.index)
             return data.join(dummies)

         def substitute_Cathedral(data):
             """
             Input:
               data (data frame): a data frame containing a 'Roof Material' column.  Its values
                                  should be limited to those found in the codebook
             Output:
               data frame identical to the input except with a refactored 'Roof Material' column
             """
             replacements = {
                 'Cathedral Ceiling': {
```

```
                1: int('1'),
                2: int('0')


            }
        }

        data = data.replace(replacements)
        return data
```

```
In [48]: def process_data_m2(data):

        # You should start by only keeping values with Pure Market Filter = 1
        repair_condition = data[['Cathedral Ceiling']]
        data = process_data_candidates(data)
        data = ohe_roof_material(data)
        data['Cathedral Ceiling'] = repair_condition
        return data






    # Process the data for Model 2
    processed_train_m2 = process_data_m2(train)

    processed_val_m2 = process_data_m2(valid)


    # Create X (dataframe) and Y (series) to use in the model
    X_train_m2 = processed_train_m2[['Log Building Square Feet', 'Roof Material_Other', 'Roof Mate
                                     'Roof Material_Slate', 'Roof Material_Tar&Gravel', 'Roof Mate
    Y_train_m2 = processed_train_m2['Log Sale Price']

    X_valid_m2 = processed_val_m2[['Log Building Square Feet', 'Roof Material_Other', 'Roof Materi
                                     'Roof Material_Slate', 'Roof Material_Tar&Gravel', 'Roof Mate
    Y_valid_m2 = processed_val_m2['Log Sale Price']


    # Take a look at the result
    display(X_train_m2.head())
    display(Y_train_m2.head())

    display(X_valid_m2.head())
    display(Y_valid_m2.head())

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

```
        Log Building Square Feet  Roof Material_Other  Roof Material_Shake  \
130829                  7.870166                  0.0                  0.0
193890                  7.002156                  0.0                  0.0
```

```
30507              6.851185                0.0                 0.0
91308              7.228388                0.0                 0.0
131132             7.990915                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  Cathedral Ceiling
130829                       0.0                 0.0                0.0
193890                       0.0                 0.0                0.0
30507                        0.0                 0.0                2.0
91308                        0.0                 0.0                0.0
131132                       0.0                 0.0                2.0


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  Roof Material_Other  Roof Material_Shake  \
50636              7.310550                0.0                 0.0
82485              7.325808                0.0                 0.0
193966             7.090077                0.0                 0.0
160612             7.281386                0.0                 0.0
7028               7.118016                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  Cathedral Ceiling
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                2.0
7028                         0.0                 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 [49]: # 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_m2 = lm.LinearRegression(fit_intercept=False)

         linear_model_m2.fit(X_train_m2, Y_train_m2)
         # your code above this line to create regression model for Model 2

         Y_predict_train_m2 = linear_model_m2.predict(X_train_m2)

         Y_predict_valid_m2 = linear_model_m2.predict(X_valid_m2)

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

```
In [50]: training_error_log[1] = rmse(Y_predict_train_m2, Y_train_m2)
         validation_error_log[1]= rmse(Y_predict_valid_m2, Y_valid_m2)

         # Training and test errors for the model (in its original values before the log transform)
         training_error[1] = rmse(np.exp(Y_predict_train_m2), np.exp(Y_train_m2))
         validation_error[1] = rmse(np.exp(Y_predict_valid_m2), np.exp(Y_valid_m2))


         print("2nd Model\nTraining RMSE (log): {}\nValidation RMSE (log): {}\n".format(training_error_l
         print("2nd Model \nTraining RMSE: {}\nValidation RMSE: {}\n".format(training_error[1], validati

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

```
2nd Model
Training RMSE (log): 0.7444323867860118
Validation RMSE (log): 0.7438500444909443

2nd Model
Training RMSE: 239251.46007045562
Validation RMSE: 244527.58972129214
```

```
In [51]: ...

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

```
Out[51]: Ellipsis
```

```
In [52]: # 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 substitute_Basement(data):
             """
             Input:
               data (data frame): a data frame containing a 'Roof Material' column.  Its values
                                  should be limited to those found in the codebook
             Output:
               data frame identical to the input except with a refactored 'Roof Material' column
             """
             replacements = {
                 'Basement': {
                     1: 'Full',
                     2: 'Slab',
                     3: 'Partial',
                     4: 'Crawl'

                 }
             }

             data = data.replace(replacements)
             return data

         def ohe_Basement(data):
             """
             One-hot-encodes roof material.  New columns are of the form x0_MATERIAL.
             """
             ...
             # BEGIN SOLUTION NO PROMPT
             oh_enc = OneHotEncoder()
             oh_enc.fit(data[['Basement']])
             dummies = pd.DataFrame(oh_enc.transform(data[['Basement']]).todense(),
                                    columns=oh_enc.get_feature_names_out(),
                                    index = data.index)
             return data.join(dummies)

         # 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
             data = data[data["Pure Market Filter"]==1]

             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 Bedrooms
    data = add_total_bedrooms(data)



    data = substitute_roof_material(data)
    data = ohe_roof_material(data)
    data = substitute_Basement(data)
    data = ohe_Basement(data)

    data = data[['Log Building Square Feet',  'Roof Material', 'Bedrooms', 'Log Sale Price',\
                 'Roof Material_Other', 'Roof Material_Shake', 'Roof Material_Shingle/Asphalt',
                 'Roof Material_Slate', 'Roof Material_Tar&Gravel', 'Roof Material_Tile',\
                 'Basement_Full', 'Basement_Slab', 'Basement_Partial', 'Basement_Crawl']]



    return 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[['Log Building Square Feet', 'Roof Material_Other', 'Roof Mate
                                 'Roof Material_Slate', 'Roof Material_Tar&Gravel', 'Roof Mate
                                 'Basement_Full', 'Basement_Slab', 'Basement_Partial', 'Baseme
Y_train_m3 = processed_train_m3["Log Sale Price"]

X_valid_m3 = processed_val_m3[['Log Building Square Feet', 'Roof Material_Other', 'Roof Materi
                                 'Roof Material_Slate', 'Roof Material_Tar&Gravel', 'Roof Mate
                               'Basement_Full', 'Basement_Slab', 'Basement_Partial', 'Basement
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  Roof Material_Other  Roof Material_Shake  \
130829                  7.870166                  0.0                  0.0
193890                  7.002156                  0.0                  0.0
30507                   6.851185                  0.0                  0.0
91308                   7.228388                  0.0                  0.0
131132                  7.990915                  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  Basement_Full  \
130829                       0.0                 0.0            1.0
193890                       0.0                 0.0            0.0
30507                        0.0                 0.0            1.0
91308                        0.0                 0.0            1.0
131132                       0.0                 0.0            0.0

        Basement_Slab  Basement_Partial  Basement_Crawl
130829            0.0               0.0             0.0
193890            1.0               0.0             0.0
30507             0.0               0.0             0.0
91308             0.0               0.0             0.0
131132            0.0               1.0             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  Roof Material_Other  Roof Material_Shake  \
50636                   7.310550                  0.0                  0.0
82485                   7.325808                  0.0                  0.0
193966                  7.090077                  0.0                  0.0
160612                  7.281386                  0.0                  0.0
7028                    7.118016                  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  Basement_Full  \
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            1.0

        Basement_Slab  Basement_Partial  Basement_Crawl
```

```
50636              1.0              0.0             0.0
82485              1.0              0.0             0.0
193966             1.0              0.0             0.0
160612             0.0              0.0             1.0
7028               0.0              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 [53]: # 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)
```

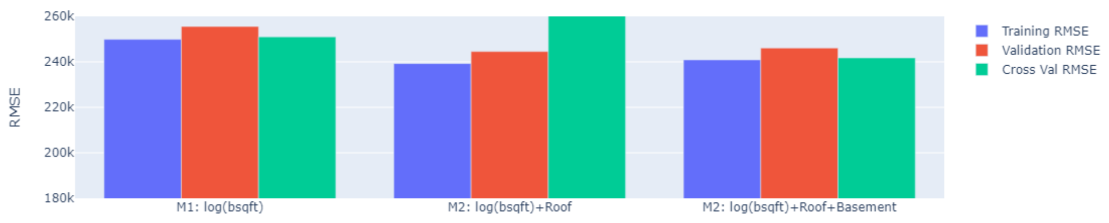
```
In [54]: # MODELING STEP 3:  Evaluate the RMSE for your model

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

```
3rd Model
Training RMSE (log): 0.7407176970355965
Validation RMSE (log): 0.7413666361194626

3rd Model
Training RMSE: 240857.53882887578
Validation RMSE: 246061.37271786112
```

```
In [55]: # 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[['Log Building Square Feet', 'Roof Material_Other', 'Roof Materi
                                         'Roof Material_Slate', 'Roof Material_Tar&Gravel', 'Roof Mater
                                         'Basement_Full', 'Basement_Slab', 'Basement_Partial', 'Basemen
         Y_full_m3 = processed_full_m3['Log Sale Price']
         # 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]))
```

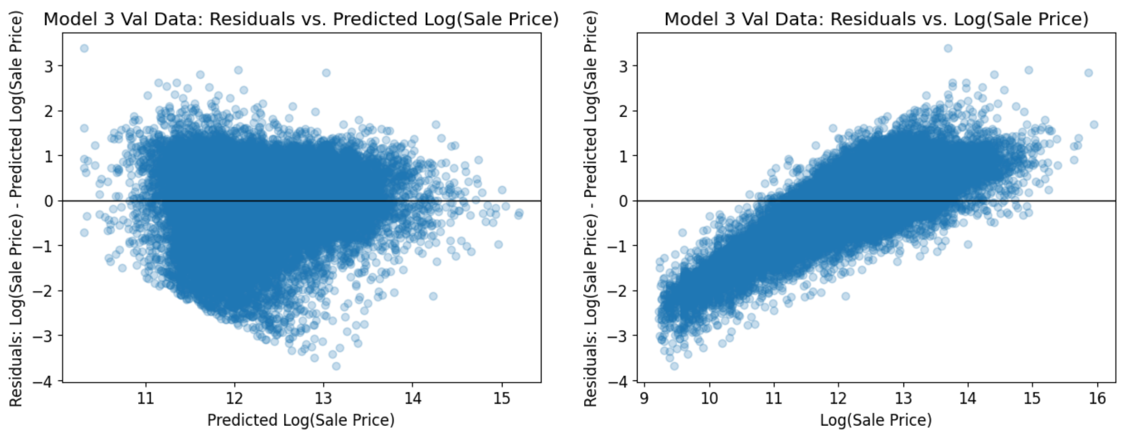
```
3rd Model Cross Validation RMSE: 241729.07227527228
```

```
In [56]: # MODELING STEP 5:  Add a name for your 3rd model describing the features and run this cell to

         model_names[2] = "M2: log(bsqft)+Roof+Basement"


         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 [57]: # 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))


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



0.1.3 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?

5a.

I chose "Basement" as a feature for Model3 because Basement type effects house price a lot.

5c.

i). Comment on your RMSE and residual plots from Model 3 compared to the first 2 models. Model 3 could make bigger difference on RMSE than Model 2 but that is still not dramatical decrease, and also the residuals look nearly the same as Model 2

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? Yes, I will need to try to break the pattern of the plot so that the plot looks spread randomly without any patterns.

iii). If you had more time to improve your model, what would your next steps be? I would use the Basement Finish attribute to add more details on the basement part.

0.2 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?

6a.

Residual value refers to the value of a fixed asset after that asset has fully depreciated. In real estate, a property's residual value would be its value at the end of these lifespans. Therefore, if a house had high residual value, high tax rates would be applied, vice versa, if a hause had low residual value, low tax rates would be applied.

As we could see CCAO Dataset, wealthier people's properties have positive residual, which means the predicted values are too low because they could lower the initial predicted values by disputing the values and also the evaluating system has discriminal factors against homeowners who leave in certain areas. Whereas people who couldn't have chances to challenge the initial predicted values and live in certain areas where the evaluating system adds penalty points have negative residual, which means the predicted values are too high.

6b.

RMSE is a good measur of accuracy but that is only good for comparing forecasting errors of different models. Even if a model had a lower RMSE than different models, which means more accurate, that doesn't mean the model is necessarilly fair because if the model still has a certain pattern with low RMSE, that can not be a fair model.

0.3 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 [ ]: # Modeling Step 1:  Process the Data




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


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


        def process_data_ec(data):

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

            return data


        # Process the data
        processed_train_ec = ...

        processed_val_ec = ...


        X_train_ec = ...
        Y_train_ec = ...

        X_valid_ec = ...
        Y_valid_ec = ...


        # 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())
```

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

        # If you are are incorporating one-hot-encoded data, set the fit_intercept to False
        ...
        # your code above this line to create regression model for Model 2

        Y_predict_train_ec = ...

        Y_predict_valid_ec = ...
```

```
In [ ]: # 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 = ...
        validation_error_ec = ...


        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, valid
```

```
In [ ]: # 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=[140000,260000], title="RMSE")
```

```
In [ ]: # 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_plt1 = ...

        x_plt2 = ...
        y_plt2 = ...


        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)")
```

0.4 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?

Type your answer here, replacing this text.