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.

For A, this may be considered unfair since it might increase the owners, property tax.

For B, this may be considered unfair as the owner might benefit from lower property tax, and is unequal distribution of taxes.

For C, this is unfair as it might contribute to a decrease in home-values, and the houses value may be questionable. This might contribute to economic inequality as well. Furthermore, it has higher taxes on properties with less value.

For D, this unfair because less expensive properties are consistently undervalues, which leads to owners of more expensive properties paying less in property taxes than their actual value of homes.

Overall, the if the assessment of the worth isn't correct, it might increase/decrease property taxes unfairly for everyone.

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).**

There were a couple problems with the property tax system in Cook County.There was racial discrimination and inequity, innaccurate residential assessments, regressive taxation, and lack of transparency and accountability. The Chicago Tribune reported the tax system in Cook County had a regressive tax system. The primary causes would be that there were assessment process that undervalues high priced homes owned by wealther people while overvaluing lower priced homes, which contributed to racial discrimination. We see that the assessment was flawed, and it affected property tax, causing unfairness in the tax system. This lead to regression, which favored weathier people, rather than working-class individuals. Furthermore, there was a lack of transparency, in which made it harder for property owners to prove that their assessments were wrong. Furthermore, a big cause was the appeal systeam. If you decide to assess your home, and CCAO valued it, if it isn't to your liking you can go to court. However, poorer people couldn't afford good tax lawyers while wealtheir people could. Thus, the poor people had their houses overvalued, with property tax increased, and couldn't win in court because they couldn't afford a lawyer. Another problem could be that people viewed it towards racism due to Jim Crow laws, and how people thought about minorities in general.

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)

We see that there are 0 N/A or negative/0 values in the data.

2aii) There are some very large values, like 71,000,000. We can cutoff this around 0.99 percent, which will reduce outliers 2aiii) We do have many small outliers, like $1. Thus we should set a cutoff around 5,000. This may be reasonable.

1 your code exploring Sale Price above this line

```
In [9]: miss = training_val_data['Sale Price'].isnull().sum()
        print("Number of miss: ", miss)

        negative_or_zero = (training_val_data['Sale Price'] <= 0).sum()
        print("Neg/Zero: ", negative_or_zero)
```

```python
# optional extra cell for exploring code

# plt.figure(figsize=(12, 6))
# sns.boxplot(x=training_val_data['Sale Price'])
# plt.show()
large = training_val_data['Sale Price'].quantile(0.999977685479)
small = training_val_data['Sale Price'].quantile(0.1771)
print("Cut off for large: ", large, "\n Cut off for small: ", small)
```

```
Number of miss:  0
Neg/Zero:  0
Cut off for large:  10000000.569297252
 Cut off for small:  5000.0
```

1.0.1 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).

```
In [40]: only_num = training_val_data.select_dtypes(include='number')
         correlation_btwn_sale = only_num.corr()['Sale Price'].drop(index=['Sale Price']).sort_values(a
         print(correlation_btwn_sale)


         # We see that the correlation between Sale Price with Fireplaces because the correlation was r
```

```
Estimate (Building)           0.609286
Estimate (Land)               0.523583
Building Square Feet          0.520472
Fireplaces                    0.395262
Pure Market Filter            0.314310
Latitude                      0.311347
Property Class                0.244005
Central Air                   0.234717
Roof Material                 0.195681
Cathedral Ceiling             0.194849
Garage 1 Size                 0.182971
Design Plan                   0.180211
Garage 1 Material             0.171696
Lot Size                      0.114456
Land Square Feet              0.114456
Most Recent Sale              0.104401
Wall Material                 0.070290
Garage Indicator              0.070130
Garage 2 Material             0.046376
Attic Type                    0.040380
Garage 2 Attachment           0.034886
Other Heating                 0.034196
Porch                         0.025360
Floodplain                    0.017337
Other Improvements            0.017241
Sale Half of Year             0.016788
Garage 2 Area                 0.016270
Sale Quarter of Year          0.013531
Sale Month of Year            0.011575
Central Heating               0.010067
Road Proximity                0.008984
O'Hare Noise                  0.004408
Number of Commercial Units    0.004063
Garage 1 Area                 0.002316
Apartments                   -0.000399
Multi Code                   -0.002529
Multi Property Indicator     -0.008140
Garage 2 Size                -0.024392
```

```
Attic Finish                  -0.036228
Basement                      -0.038668
Garage 1 Attachment           -0.042414
Deed No.                      -0.047138
Sale Half-Year                -0.047139
Sale Quarter                  -0.047404
Sale Year                     -0.049317
Site Desirability             -0.065321
Census Tract                  -0.066156
Repair Condition              -0.086370
Town Code                     -0.089018
Longitude                     -0.110532
Neighborhood Code             -0.118884
Neigborhood Code (mapping)    -0.118884
Basement Finish               -0.129076
Construction Quality          -0.132195
Town and Neighborhood         -0.144724
Age Decade                    -0.180765
Age                           -0.180765
PIN                           -0.299936
Use                                 NaN
Name: Sale Price, dtype: float64
```

In [41]:
```
...

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

Out[41]: Ellipsis

In [42]:
```
...

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

Out[42]: Ellipsis

In [43]:
```
...

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

Out[43]: Ellipsis

In [44]:
```
# 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 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'])
    data['Log Building Square Feet'] = np.log(data['Building Square Feet'])
    ohe = ohe_roof_material(data)
    model_columns = [i for i in ohe if i.startswith('Roof Material_')] + ['Log Building Square
    return ohe[model_columns + ['Log Sale Price']]


    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.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())
```

```
        Roof Material_1.0  Roof Material_2.0  Roof Material_3.0  \
130829                1.0                0.0                0.0
193890                1.0                0.0                0.0
30507                 1.0                0.0                0.0
91308                 1.0                0.0                0.0
```

```
131132               1.0                0.0                0.0

        Roof Material_4.0  Roof Material_5.0  Roof Material_6.0  \
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

        Log Building Square Feet  Fireplaces
130829                  7.870166         1.0
193890                  7.002156         0.0
30507                   6.851185         0.0
91308                   7.228388         0.0
131132                  7.990915         1.0
```

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

```
        Roof Material_1.0  Roof Material_2.0  Roof Material_3.0  \
50636                 1.0                0.0                0.0
82485                 1.0                0.0                0.0
193966                1.0                0.0                0.0
160612                1.0                0.0                0.0
7028                  1.0                0.0                0.0

        Roof Material_4.0  Roof Material_5.0  Roof Material_6.0  \
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

        Log Building Square Feet  Fireplaces
50636                   7.310550         0.0
82485                   7.325808         0.0
193966                  7.090077         0.0
160612                  7.281386         0.0
7028                    7.118016         0.0
```

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

In [45]:
```
# 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 [46]:
```
# 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
```

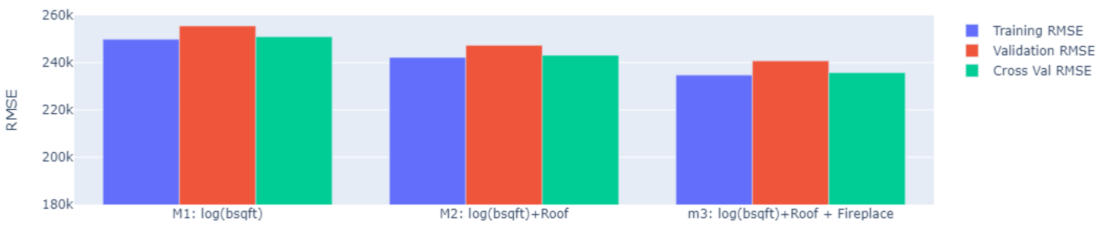
```
3rd Model
Training RMSE (log): 0.7398769188124334
Validation RMSE (log): 0.7396234774188858

3rd Model
Training RMSE: 234735.8421357366
Validation RMSE: 240806.5380592281
```

In [47]:
```
# 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)


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"]

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: 235803.57775227673
```

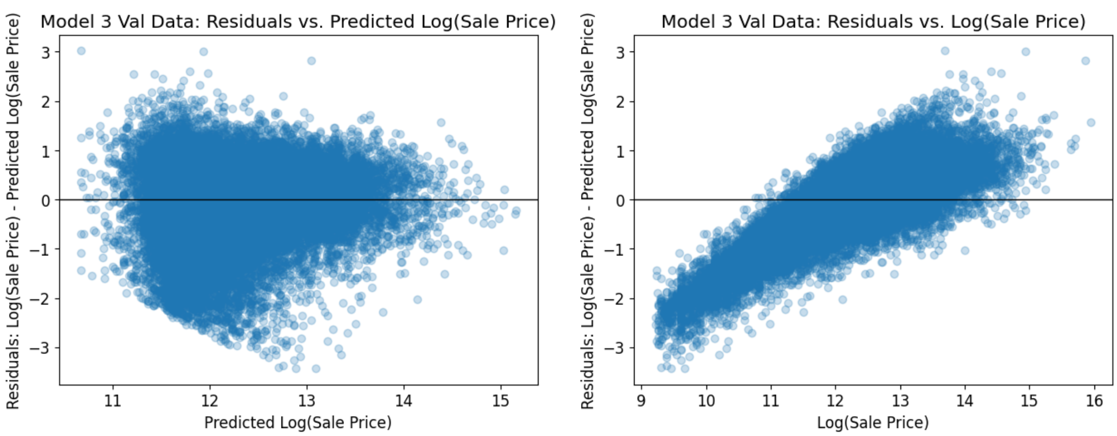
In [48]:
```
# MODELING STEP 5:  Add a name for your 3rd model describing the features and run this cell to

model_names[2] = "m3: log(bsqft)+Roof + Fireplace"



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 [49]:
```
# 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[49]: Text(0.5, 1.0, 'Model 3 Val Data: Residuals vs. Log(Sale Price)')



1.0.2 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) We wee that the RMSE decreases as we add more features. This shows that we have a more accurate model. Furthermore, we can see that the residuals are beginning to randomize even more.

ii) The residuals in the model is still showing a trend. We see that the trend still overestimates lower priced houses. We could add another feature with a higher correlation, and use transformations.

iii) My next steps would be to find better correlations of the model and try to improve the plot, and try to decrease the RMSE with fireplaces data.

1.1 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?

A model's prediction of property values for tax assessments can be 'fair' in many ways. I believe that its important to have as much clean data as we can, and keep updating the data to match its actual home value. Fair isn't based off of RMSE, and having a low RMSE doesn't mean its fair. If the data that we have is skewed, then many problems affecting people. Fairness is when everyone gets equal treatment, and not how much we can predict to the actual value through these data analysis.

1.2 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 [68]: # 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 ohe_roof_material(data):
             """
             One-hot-encodes roof material.  New columns are of the form "Roof Material_MATERIAL"
             """
             ohe_Roof = OneHotEncoder()
             ohe_Roof_columns = ohe_Roof.fit_transform(data[['Roof Material']])
             ohe_Roofdf = pd.DataFrame(ohe_Roof_columns.todense(),
                                       columns=ohe_Roof.get_feature_names_out(['Roof Material']),
                                       index=data.index)
             one_Roofdata = pd.merge(data, ohe_Roofdf, left_index=True, right_index=True)
             return one_Roofdata
         # Optional:  Define any helper functions you need (for example, for one-hot encoding, etc) abo


         def process_data_ec(data, is_test_set=False):
             # Please include all of your feature engineering processes for both the training/validatio
             # Can include feature engineering processes for both the training/validation and the test



             # 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['Log Sale Price'] = np.log(data['Sale Price'])
                 data = data[data["Pure Market Filter"]==1]
                 data['Log Building Square Feet'] = np.log(data['Building Square Feet'])
                 data['Log Estimate (Building)'] = np.log(data['Estimate (Building)'])
                 data = data.replace([np.inf, -np.inf], np.log(data['Estimate (Building)'].mean()))
             #     From the describe, we see that it was -infy since we took the log(0). Thus we need t
                 ohe = ohe_roof_material(data)
                 model_columns = [i for i in ohe if i.startswith('Roof Material_')] + ['Log Building Squ
                 print(data['Log Estimate (Building)'].describe(), '\n')
                 # Include the rest of your feature engineering processes for the training/validation s
```

```
                return ohe[model_columns + ['Log Sale Price']]

            else:
                # Processing for the test set
                # CANNOT involve references to sale price!
                # CANNOT involve removing any rows
                data['Log Building Square Feet'] = np.log(data['Building Square Feet'])
                data['Log Estimate (Building)'] = np.log(data['Estimate (Building)'])
                data = data.replace([np.inf, -np.inf], np.log(data['Estimate (Building)'].mean()))
                ohe = ohe_roof_material(data)
                model_columns = [i for i in ohe if i.startswith('Roof Material_')] + ['Log Building Squ
                print(data['Log Estimate (Building)'].describe(), '\n')
                return ohe[model_columns]


            # Add any remaining processing for both test and training set below (hint - easiest to put

            return 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())
```

```
count    133849.000000
mean         12.021993
std           0.717310
min           5.075174
25%          11.545780
50%          12.009206
75%          12.432810
max          15.589644
Name: Log Estimate (Building), dtype: float64

count    33535.000000
```

```
mean        12.024355
std          0.715668
min          6.109248
25%         11.539470
50%         12.006829
75%         12.440710
max         15.830087
Name: Log Estimate (Building), dtype: float64
```

```
        Roof Material_1.0  Roof Material_2.0  Roof Material_3.0  \
130829                1.0                0.0                0.0
193890                1.0                0.0                0.0
30507                 1.0                0.0                0.0
91308                 1.0                0.0                0.0
131132                1.0                0.0                0.0

        Roof Material_4.0  Roof Material_5.0  Roof Material_6.0  \
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

        Log Building Square Feet  Fireplaces  Log Estimate (Building)
130829                  7.870166         1.0                13.019932
193890                  7.002156         0.0                10.969749
30507                   6.851185         0.0                11.569589
91308                   7.228388         0.0                12.839682
131132                  7.990915         1.0                12.357548
```

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

```
        Roof Material_1.0  Roof Material_2.0  Roof Material_3.0  \
50636                 1.0                0.0                0.0
82485                 1.0                0.0                0.0
193966                1.0                0.0                0.0
160612                1.0                0.0                0.0
7028                  1.0                0.0                0.0

        Roof Material_4.0  Roof Material_5.0  Roof Material_6.0  \
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

        Log Building Square Feet  Fireplaces  Log Estimate (Building)
50636                   7.310550         0.0                11.669758
82485                   7.325808         0.0                12.264672
193966                  7.090077         0.0                10.669885
160612                  7.281386         0.0                12.154095
7028                    7.118016         0.0                11.355570


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

In [69]:
```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 [70]:
```python
# MODELING STEP 3:  Evaluate the RMSE for your model


training_error_log_ec = rmse(Y_predict_train_ec, Y_train_ec)
validation_error_log_ec= rmse(Y_predict_valid_ec, Y_valid_ec)
# 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, vali
```

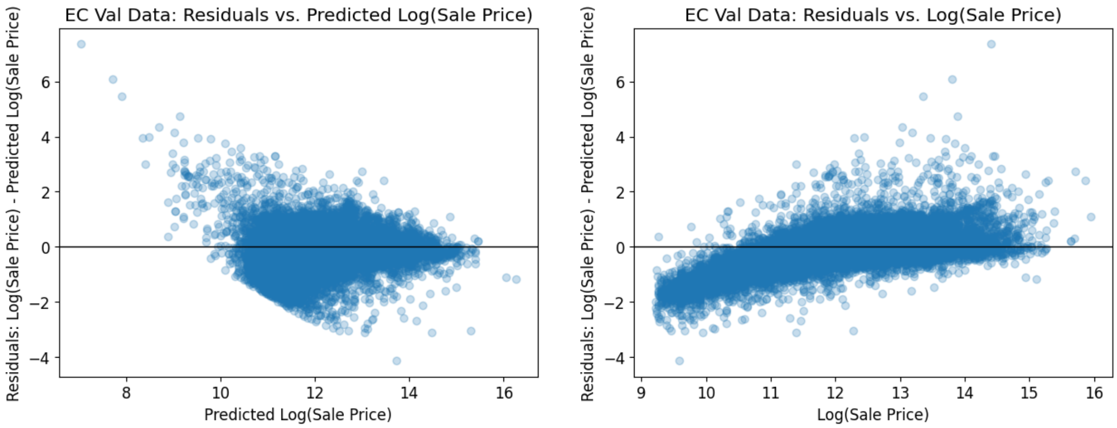
```
Extra Credit
Training RMSE: 183994.41486187957
Validation RMSE: 196717.60467890708
```

In [71]:
```python
# 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")
# Feel free to update the range as needed
```

In [72]:
```
# 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[72]: Text(0.5, 1.0, 'EC Val Data: Residuals vs. Log(Sale Price)')



1.3 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?

To create our model, I first tried to just do 'Log Estimate (Building)'. However, the values were very big. Thus we took the log of these values. But there were values in this which included 0. But log(0) is -infinity, which won't run in the Linear Regression. Thus we need a way to replace these values. So, I made 0 or -infity values to Nan, then dropped Nan afterwards. This way we don't replace/delete rows but only the values. Now that the values that were problamatic were removed, we used the model to train on a cleaner dataset. The RMSE valuues: training RMSE is 173506.62071841123, and the validation RMSE is 185094.200492451. Thus we see that the model works, and the RMSE is below 200k. We see that these values is making more reasonable and accurate predictions than before. The trend is more randomized around 0, but does show a bit of a pattern, but it is very randomized compared to the previous 3 models above.