PRESENTED BY: WARREN MORELLI

EVALUATING & FORECASTING PROPERTY PRICES THROUGH MODELLING

10 JAN, 2023

AGENDA OUTLINE

THE FOLLOWING TOPICS WILL BE COVERED IN THIS PRESENTATION:

1. BUSINESS PROBLEM
2. OBTAINING DATA
3. DATA PREPARATION
4. MODELLING
5. EVALUATION
6. DEPLOYMENT - PREDICTIVE MODELING SOLUTION
7. NECESSARY REVISIONS TO ENHANCE MODEL ACCURACY

BUSINESS PROBLEM

A Real Estate Buyer's Agency is seeking to identify key property features, such as square footage and number of rooms, to identify undervalued properties that can be presented as investment opportunities for clients.

OBTAINING DATA

All data was imported from the kc_house_data CSV file.

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_above	sqft_basement	yr_built	yr_ren
0	7129300520	10/13/2014	221900.0	3	1.00	1180	5650	1.0	NaN	0.0	...	7	1180	0.0	1955	
1	6414100192	12/9/2014	538000.0	3	2.25	2570	7242	2.0	0.0	0.0	...	7	2170	400.0	1951	
2	5631500400	2/25/2015	180000.0	2	1.00	770	10000	1.0	0.0	0.0	...	6	770	0.0	1933	
3	2487200875	12/9/2014	604000.0	4	3.00	1960	5000	1.0	0.0	0.0	...	7	1050	910.0	1965	
4	1954400510	2/18/2015	510000.0	3	2.00	1680	8080	1.0	0.0	0.0	...	8	1680	0.0	1987	

5 rows × 21 columns

This data set contains 21,597 rows × 21 columns. A suitable amount of data for modelling.

The columns include: id, date, price, bedrooms, bathrooms, sqft_living, sqft_lot, floors, waterfront, view, grade, sqft_above, sqft_basement, yr_built, yr_renovated, zipcode, lat, long

DATA PREPARATION - CLEANING

To ensure the data was suitable for further analysis and modelling, the following steps were taken to clean and prepare the data after initial inspection:

DROP UNNECESSARY COLUMNS

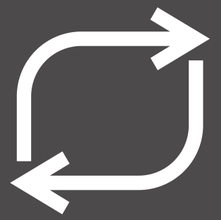
I deleted the following columns from the data set as they were not crucial to my primary objective of identifying undervalued properties: date, zip code, latitude, identifier, longitude, year of renovation, and basement size.

DROP OUTLIERS

During my examination of the data, I discovered that Bedroom and lot size contained outliers which required attention. Upon careful consideration, I determined that the most effective solution was to eliminate these outliers entirely.

DROP NULL VALUES

During my analysis of the data, I encountered a minimal percentage of null values. After evaluating the impact of these null values on the overall outcome, I made the informed decision to remove them from the data set.



CONVERT DATA TYPE

Several columns were found to have a data type of "string," which would result in issues during the modeling process. To mitigate this, I converted these columns to a data type of "integer."

DATA PREPARATION - CLEANING

Upon inspecting the data, I implemented the following steps to clean and prepare it for further use:

CONVERT 'YEAR BUILT'

To facilitate modeling, I performed a transformation on the data, converting the "year built" feature to "years old." This conversion was intended to simplify the modeling process.

DATA PREPARATION
ASSESING MULTICOLLINEARITY

By utilizing Variance Inflation Factors (VIF), I assessed the data for multicollinearity. The objective is to construct a model with minimal impact from multicollinearity. To achieve this, some predictors were selectively removed while maintaining a favourable ratio of continuous to categorical variables. This was done with consideration of avoiding a dummy trap during the modelling process.

After assessing the effect of removing certain variables and observing their impact on other variables, I determined that the best course of action was to eliminate three specific attributes from the data set: grade, condition, and view.

	feature	VIF
0	bedrooms	22.290011
1	bathrooms	16.044053
2	sqft_living	43.359099
3	sqft_lot	1.231118
4	floors	15.186057
5	waterfront	1.211481
6	view	1.485341
7	condition	27.403483
8	grade	57.609510
9	sqft_above	33.835907
10	yrs_old	5.233397

DATA PREPARATION
DEALING WITH CATEGORICAL VALUES

To address the categorical predictors, I utilized the one hot encoding (OHE) method to generate dummy variables. However, this resulted in a large number of dummy variables, which could lead to the issue of the "dummy trap."

To mitigate this challenge, I divided the variables "bedrooms" and "bathrooms" into smaller categories. The variable "bedrooms" was grouped into three categories: 1-3 bedrooms, 4-7 bedrooms, and 7+ bedrooms. Meanwhile, the variable "bathrooms" was grouped into two categories: 1-3 bathrooms, and 4+ bathrooms.

	price	sqft_living	sqft_lot	floors	waterfront	sqft_above	yrs_old	4-7_bedrooms	7+_bedrooms	4+_bathrooms
1	538000.0	2570	7242.0	2	0	2170	65	0	0	0
2	180000.0	770	10000.0	1	0	770	83	0	0	0
3	604000.0	1960	5000.0	1	0	1050	51	1	0	0
4	510000.0	1680	8080.0	1	0	1680	29	0	0	0
5	1230000.0	5420	101930.0	1	0	3890	15	1	0	1

MODELLING

Step 1:
Split data into training and testing and use Cross-Validation
The data was splip into 4 sections. X_train, X_test, y_train, y_test

Step 2:
Fit a linear regression model and calculate mean squared error (MSE)
The MSE was in an acceptable range. Next, I performed a Cross-Validation to make sure this model is correct.

```
train_mse = mean_squared_error(y_train, y_hat_train)
test_mse = mean_squared_error(y_test, y_hat_test)
print('Train Mean Squared Error:', train_mse)
print('Test Mean Squared Error:', test_mse)
```

```
Train Mean Squared Error: 56790822769.71211
Test Mean Squared Error: 57763716144.224464
```

```
In [36]: dif = test_avg - train_avg
         train_ratio = dif / train_avg
         train_ratio

Out[36]: 0.00670574732186624
```

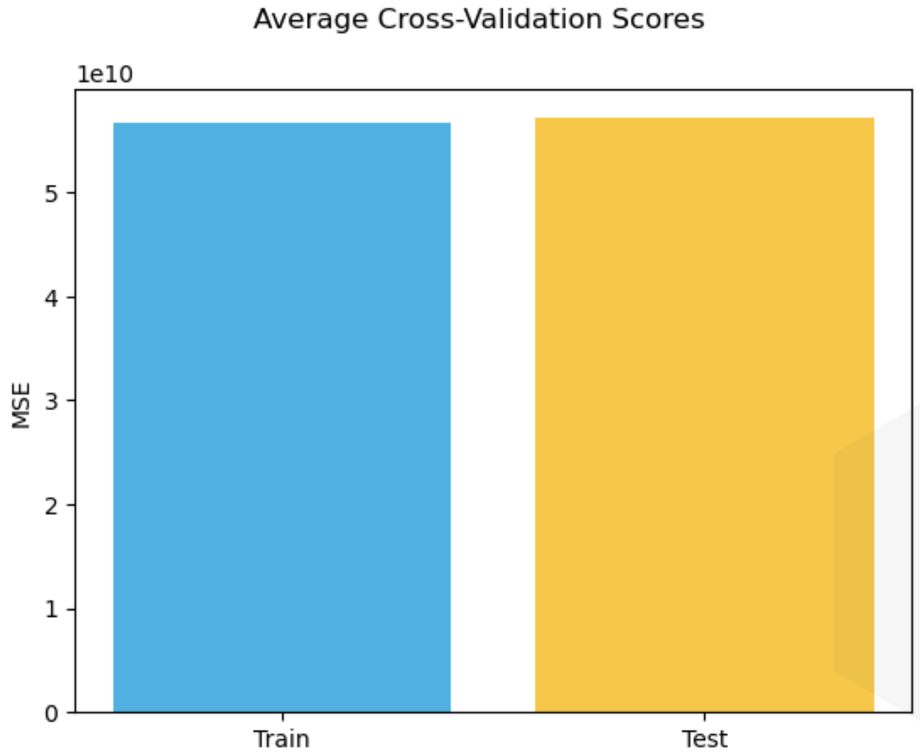
```
In [37]: test_ratio=dif/test_avg
         test_ratio

Out[37]: 0.006661079803811097
```

MODELLING

Step 3:

Perform a Cross-Validation

The results of the cross-validation showed that the mean squared errors (MSE) were close, indicating that the model's performance on the training data is consistent with its performance on the test data. This is a positive sign and suggests that the model has the potential to make accurate predictions on new, unseen data.




```
#Cross Validation MSE
print('Cross Validation Test Mean Squared Error:', test_avg)
print('Cross Validation Train Mean Squared Error:', train_avg)
```

```
Cross Validation Test Mean Squared Error: 57133527323.77781
Cross Validation Train Mean Squared Error: 56752956338.8009
```

MODELLING

Step 4:

View a Ordinary Least Squares (OLS) Model

Use OLS model to estimate the parameters of a linear regression model.

The model found that all of the predictors expect the number of floors, have strong correlation to the price of a property.

The adj. R-squared suggests that the independent variables in the linear regression model explain about 59% of the variation in the dependent variable. This is a moderate level of goodness-of-fit suggesting that the model provides a reasonable explanation for the variation in the dependent variable, but it also implies that there is still room for improvement, and that the model may not be ideal for making predictions on new data.

The coefficient values suggest waterfront properties have the biggest impact on the price.

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.586
Model:	OLS	**Adj. R-squared:**	0.586
Method:	Least Squares	**F-statistic:**	2112.
Date:	Wed, 08 Feb 2023	**Prob (F-statistic):**	0.00
Time:	17:40:34	**Log-Likelihood:**	-1.8509e+05
No. Observations:	13412	**AIC:**	3.702e+05
Df Residuals:	13402	**BIC:**	3.703e+05
Df Model:	9		
Covariance Type:	nonrobust		

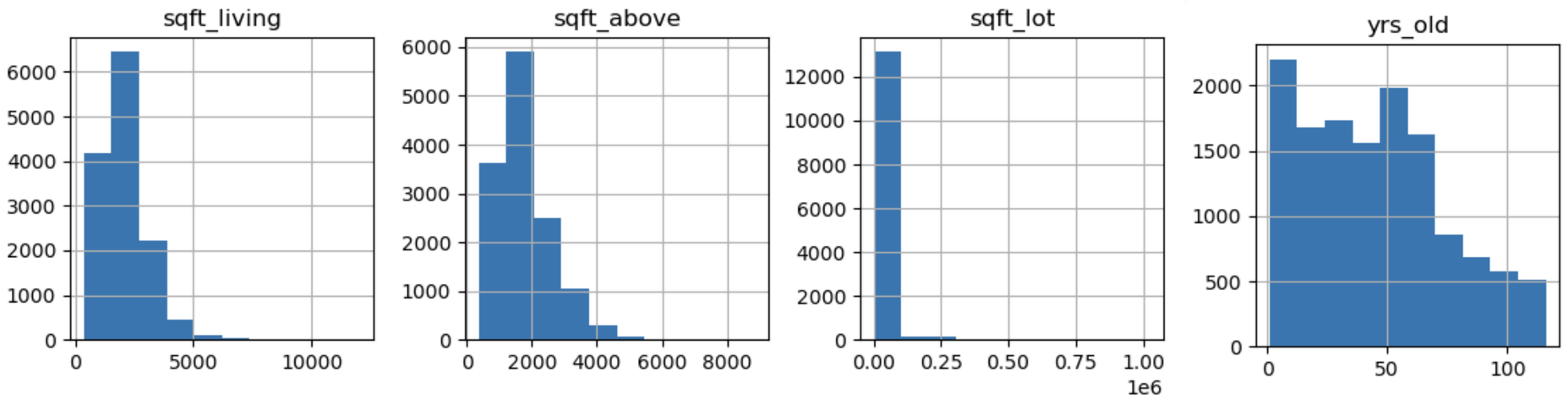
	coef	std err	t	P>|t|	[0.025	0.975]
const	-2.054e+05	7936.366	-25.875	0.000	-2.21e+05	-1.9e+05
sqft_living	277.5052	5.176	53.619	0.000	267.360	287.650
sqft_above	44.7400	6.028	7.422	0.000	32.924	56.556
sqft_lot	-0.4524	0.055	-8.283	0.000	-0.559	-0.345
yrs_old	2587.0422	84.182	30.732	0.000	2422.033	2752.051
4-7_bedrooms	-8.018e+04	5033.902	-15.929	0.000	-9.01e+04	-7.03e+04
7+_bedrooms	-2.661e+05	3.66e+04	-7.270	0.000	-3.38e+05	-1.94e+05
4+_bathrooms	2.901e+05	1.67e+04	17.420	0.000	2.57e+05	3.23e+05
floors_2	8972.1489	5776.510	1.553	0.120	-2350.624	2.03e+04
waterfront_1	7.052e+05	2.44e+04	28.909	0.000	6.57e+05	7.53e+05

Omnibus:	7377.337	**Durbin-Watson:**	1.981
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	217735.724
Skew:	2.088	**Prob(JB):**	0.00
Kurtosis:	22.292	**Cond. No.**	7.45e+05

MODELLING ITERATION 2

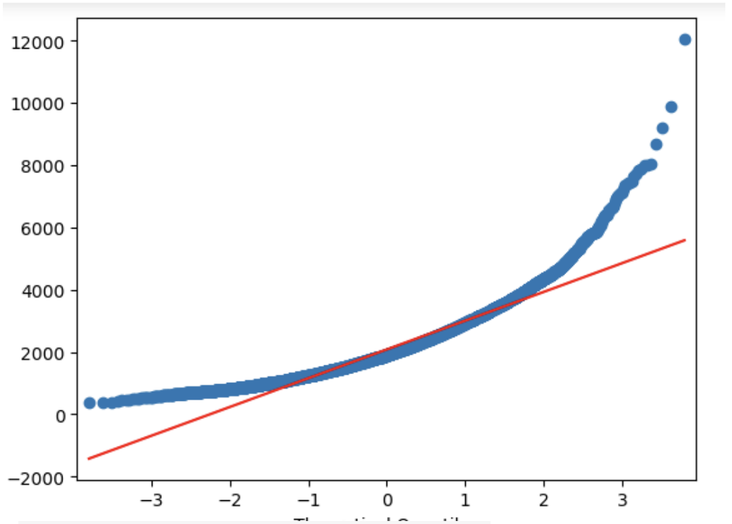
Step 1:
Delete the 'floors' variable as this doesn't effect the price of a property

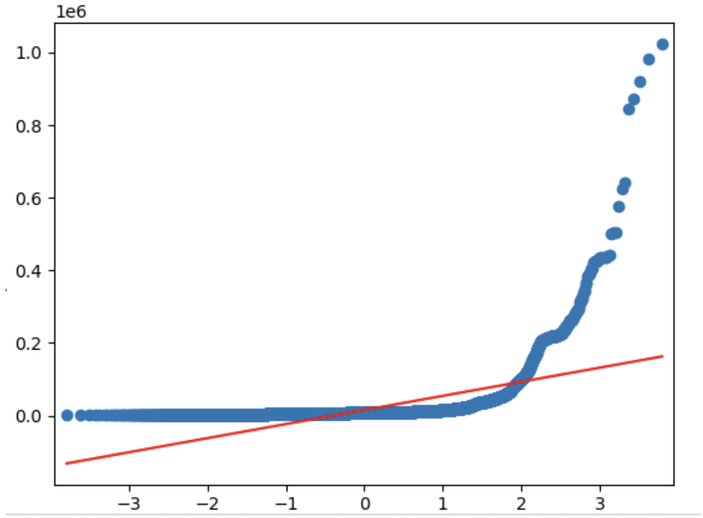
Step 2:
Check the continuous variables for skewness
I used a histogram to check the variables for Skewness to see if there was room for improvement.

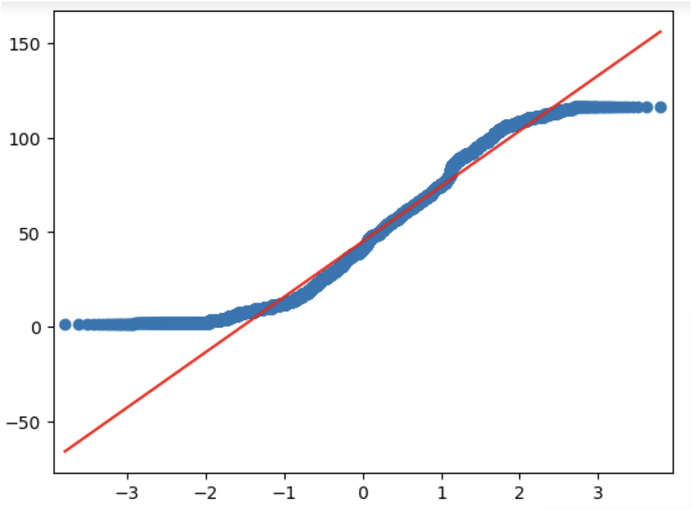


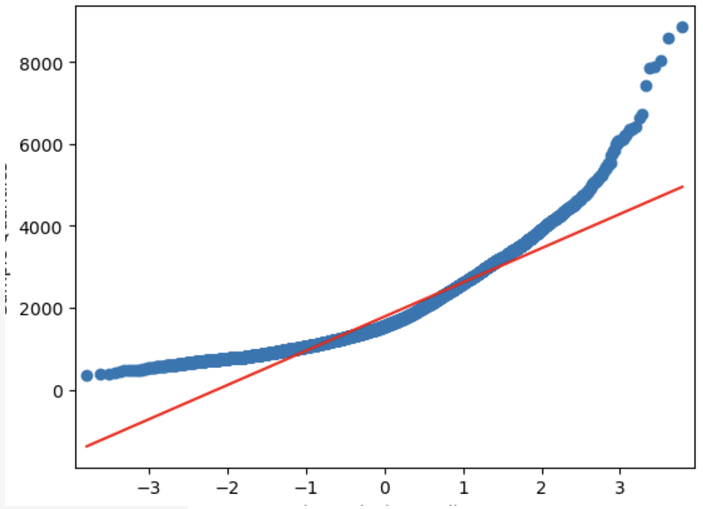
MODELLING - ITERATION 2

Step 3:
Check Q-Q Plot to confirm
The histogram suggests there is a great deal of skewness. I used a Q-Q plot to confirm this.









MODELLING - ITERATION 2

Step 4:

Use a log transformation to improve skewness and check results

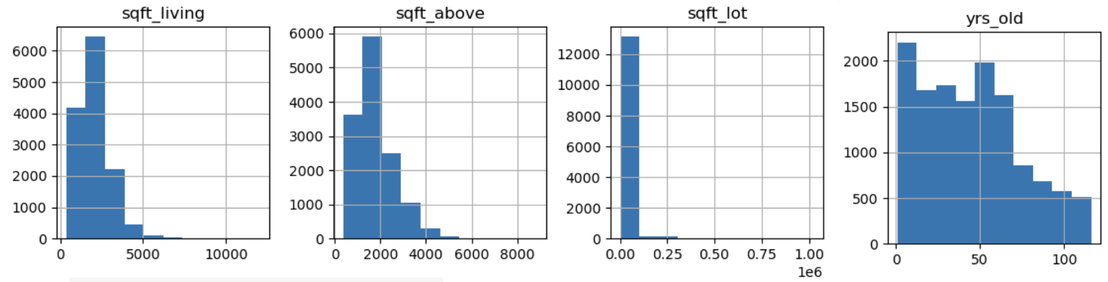
As you can see below, the log transformation greatly improved the skewness for all values except for years old.

	sqft_living	sqft_above	sqft_lot	yrs_old	4-7_bedrooms	7+_bedrooms	4+_bathrooms	waterfront_1
5576	2450	2450	4668.0	12	1	0	0	0

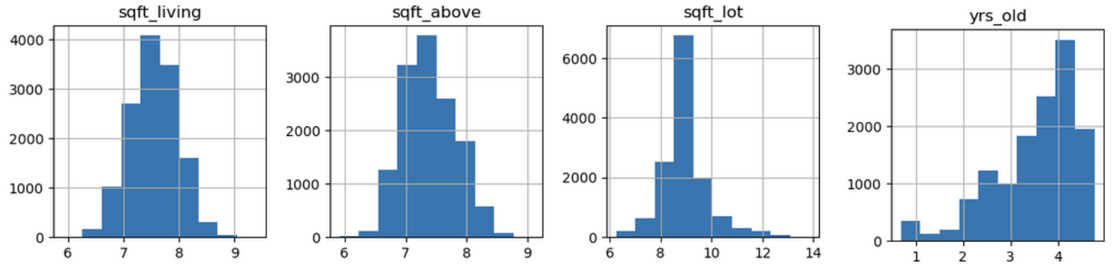
Before Log transformation

	sqft_living	sqft_above	sqft_lot	yrs_old	4-7_bedrooms	7+_bedrooms	4+_bathrooms	waterfront_1
5576	7.804251	7.804251	8.4487	2.564949	1	0	0	0

After Log transformation



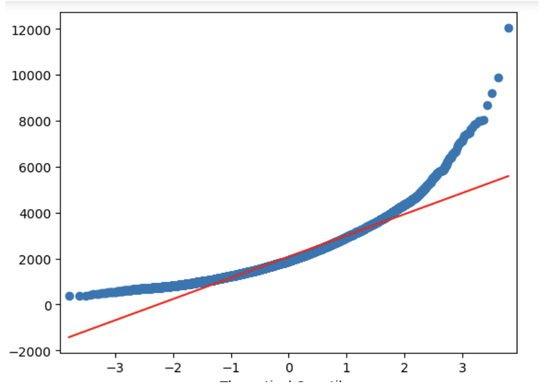
Before Log transformation

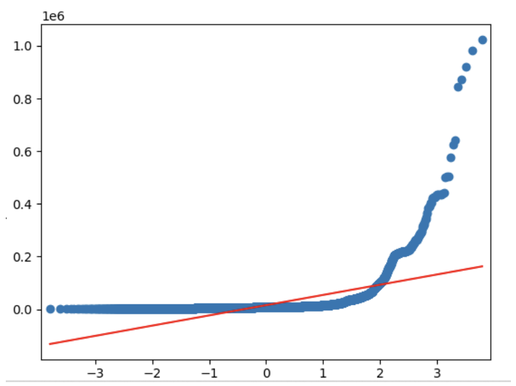


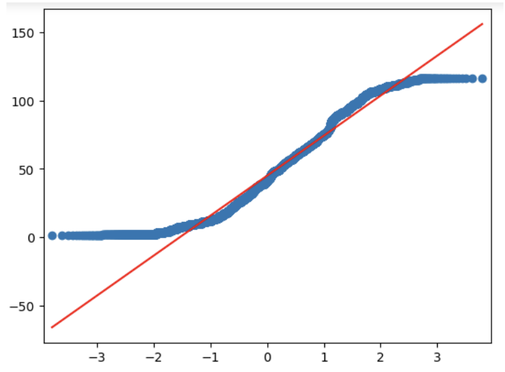
After Log transformation

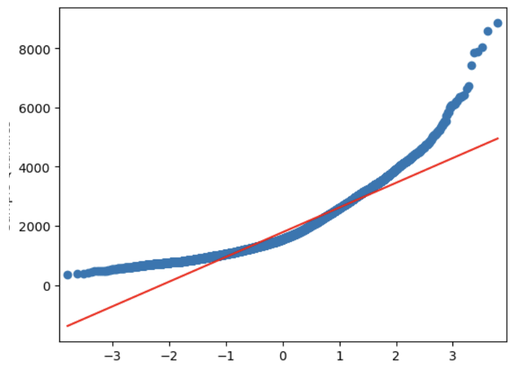
MODELLING - ITERATION 2

Before Log transformation

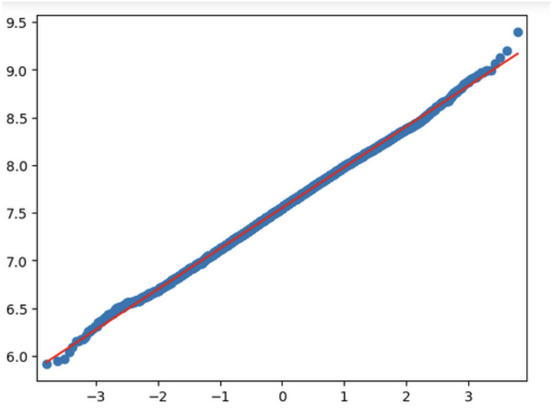


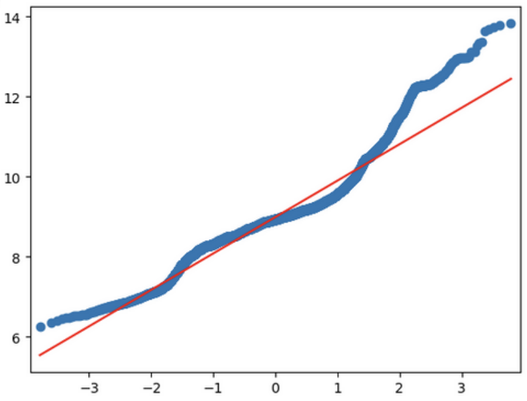


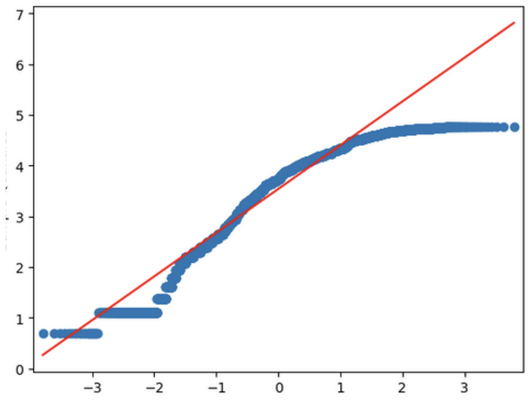


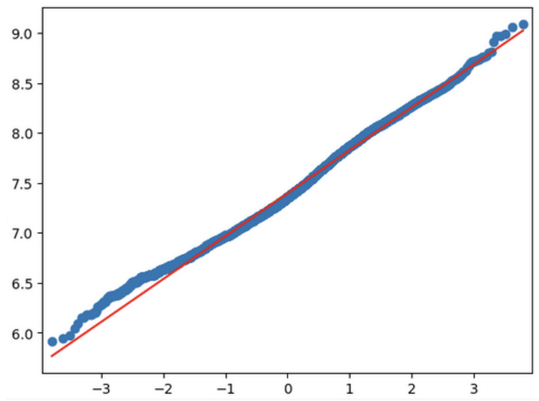


After Log transformation









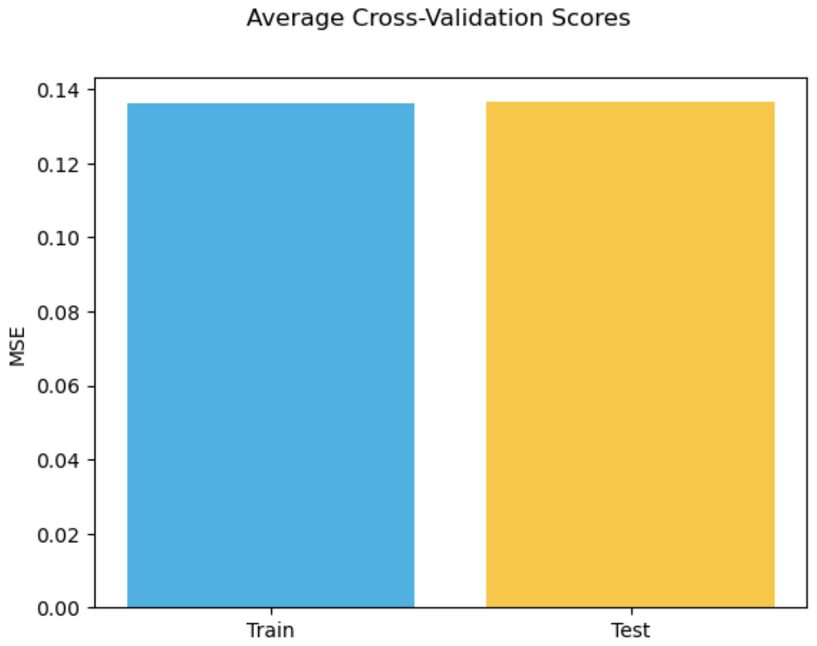
MODELLING - ITERATION 2

Step 5:

Re-check the MSE and cross-validation to ensure they are still close and the model still has the potential to make accurate predictions.

```
#calculate the MSE
train_mse = mean_squared_error(y_train, y_hat_train)
test_mse = mean_squared_error(y_test, y_hat_test)
print('Train Mean Squared Error:', train_mse)
print('Test Mean Squared Error:', test_mse)
```

```
Train Mean Squared Error: 0.13630376632999675
Test Mean Squared Error: 0.13961887507885778
```




```
#Cross Validation MSE
print('Cross Validation Test Mean Squared Error:', test_avg)
print('Cross Validation Train Mean Squared Error:', train_avg)
```

```
Cross Validation Test Mean Squared Error: 0.1364934206252361
Cross Validation Train Mean Squared Error: 0.1362827077109901
```

MODELLING - ITERATION 2

Step 6:

View the Ordinary Least Squares (OLS) Model

The model found the adj. R-squared has considerably dropped after removing the floors variable.

You can also see by the p-value that all of the variables now have a relationship with property prices

The coefficient values are more clear now suggesting waterfront properties, living space size and the number of bathrooms have the biggest impact on the overall price.

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.508
Model:	OLS	**Adj. R-squared:**	0.508
Method:	Least Squares	**F-statistic:**	1733.
Date:	Wed, 08 Feb 2023	**Prob (F-statistic):**	0.00
Time:	19:03:11	**Log-Likelihood:**	-5664.9
No. Observations:	13408	**AIC:**	1.135e+04
Df Residuals:	13399	**BIC:**	1.142e+04
Df Model:	8		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	6.0965	0.082	74.200	0.000	5.935	6.258
sqft_living	0.8222	0.016	50.432	0.000	0.790	0.854
sqft_above	0.1651	0.016	10.208	0.000	0.133	0.197
sqft_lot	-0.0885	0.004	-22.082	0.000	-0.096	-0.081
yrs_old	0.0950	0.004	21.453	0.000	0.086	0.104
4-7_bedrooms	-0.0705	0.008	-8.935	0.000	-0.086	-0.055
7+_bedrooms	-0.2380	0.061	-3.921	0.000	-0.357	-0.119
4+_bathrooms	0.3543	0.025	14.415	0.000	0.306	0.402
waterfront_1	0.6940	0.037	18.612	0.000	0.621	0.767

Omnibus:	17.661	**Durbin-Watson:**	1.979
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	15.264
Skew:	-0.021	**Prob(JB):**	0.000485
Kurtosis:	2.840	**Cond. No.**	376.

MODELLING ITERATION 3

The current low adjusted R-squared in my model may impact its predictive power. To address this, I plan to incorporate additional predictors with strong correlations to the property price, specifically the 'grade' variable.

Additionally, analysis of the previous Q-Q plot still reveals the presence of outliers that are influencing the model. To mitigate this issue, I intend to utilize a normalization technique.

Step 1:

Introduce the 'grade' variable

I constructed a new data frame incorporating the 'grade' variable and applied log transformation to align with the characteristics of the existing data. Subsequently, I merged this data frame with the x_test and x_train data frames.

```
grade_log = np.log(data_mc['grade'] + 1)
```

```
X_train = pd.concat([X_train, grade_log], axis=1)
X_test = pd.concat([X_test, grade_log], axis=1)

X_train = X_train.dropna()
X_test = X_test.dropna()
```

MODELLING ITERATION 3

Step 2:

Mean normalization

Analysis of the previous Q-Q plot still reveals the presence of outliers that are influencing the model. To mitigate this issue, I intend to utilize a mean normalization transformation. This should also give me a better interpretation of the results, as each feature is now on the same scale and can be easily compared to one another.

```
def normalize(series):
    return (series - series.mean()) / series.std()

X_train[cont] = X_train[cont].apply(normalize)
X_test[cont] = X_test[cont].apply(normalize)
y_train = normalize(y_train)
y_test = normalize(y_test)
```

MODELLING - ITERATION 3

Step 3:
Re-check OLS
Now that we should have a clearer visualisation of the OLS model, let us revisit it for accuracy.

Upon analysis, it has been determined that the variables "Square Feet Above" and "7+ Bedrooms" do not impact the sale price and will be removed from the predictors. This will not affect the adjusted R-squared, which is now demonstrating improved results.

```
X_train.drop('7+_bedrooms', inplace=True, axis=1)
X_test.drop('7+_bedrooms', inplace=True, axis=1)
X_train.drop('sqft_above', inplace=True, axis=1)
X_test.drop('sqft_above', inplace=True, axis=1)
```

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.605
Model:	OLS	**Adj. R-squared:**	0.605
Method:	Least Squares	**F-statistic:**	2282.
Date:	Thu, 09 Feb 2023	**Prob (F-statistic):**	0.00
Time:	10:44:18	**Log-Likelihood:**	-12794.
No. Observations:	13408	**AIC:**	2.561e+04
Df Residuals:	13398	**BIC:**	2.568e+04
Df Model:	9		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.0039	0.008	-0.493	0.622	-0.019	0.012
sqft_living	0.4046	0.013	32.010	0.000	0.380	0.429
sqft_above	-0.0089	0.012	-0.740	0.459	-0.033	0.015
sqft_lot	-0.1256	0.006	-20.348	0.000	-0.138	-0.114
yrs_old	0.2359	0.007	35.312	0.000	0.223	0.249
4-7_bedrooms	-0.0348	0.014	-2.571	0.010	-0.061	-0.008
7+_bedrooms	-0.0295	0.104	-0.285	0.776	-0.233	0.173
4+_bathrooms	0.4793	0.042	11.421	0.000	0.397	0.562
waterfront_1	1.2047	0.063	18.973	0.000	1.080	1.329
grade	0.5108	0.009	57.297	0.000	0.493	0.528

Omnibus:	25.980	**Durbin-Watson:**	1.998
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	31.976
Skew:	0.000	**Prob(JB):**	1.14e-07
Kurtosis:	3.239	**Cond. No.**	33.1

MODELLING - ITERATION 3

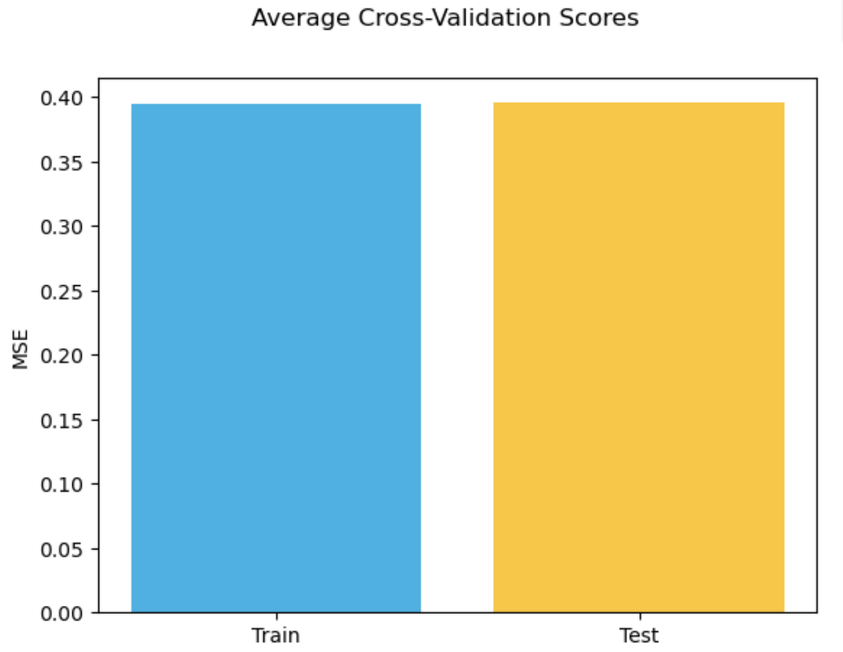
Step 4:

Final MSE check

As a final step in ensuring accuracy, I conducted one more mean squared error assessment to confirm that the model is capable of making precise predictions. Results indicate that the model will perform optimally.

```
#calculate the MSE
train_mse = mean_squared_error(y_train, y_hat_train)
test_mse = mean_squared_error(y_test, y_hat_test)
print('Train Mean Squared Error:', train_mse)
print('Test Mean Squared Error:', test_mse)
```

```
Train Mean Squared Error: 0.39479111205319956
Test Mean Squared Error: 0.40677708114208466
```




```
#Cross Validation MSE
print('Cross Validation Test Mean Squared Error:', test_avg)
print('Cross Validation Train Mean Squared Error:', train_avg)
```

```
Cross Validation Test Mean Squared Error: 0.39554550768025376
Cross Validation Train Mean Squared Error: 0.39470741163667333
```

MODELLING - ITERATION 3

Step 4:

Final OLS check

"In order to verify the model's reliability after removing the insignificant variables, I conducted a final ordinary least squares check. This confirmed that the adjusted R-squared remained within an acceptable range and no unforeseen changes had occurred.

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.605
Model:	OLS	**Adj. R-squared:**	0.605
Method:	Least Squares	**F-statistic:**	2934.
Date:	Thu, 09 Feb 2023	**Prob (F-statistic):**	0.00
Time:	10:55:24	**Log-Likelihood:**	-12794.
No. Observations:	13408	**AIC:**	2.560e+04
Df Residuals:	13400	**BIC:**	2.566e+04
Df Model:	7		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.0040	0.008	-0.517	0.605	-0.019	0.011
sqft_living	0.3984	0.010	40.619	0.000	0.379	0.418
sqft_lot	-0.1265	0.006	-21.005	0.000	-0.138	-0.115
yrs_old	0.2374	0.006	37.477	0.000	0.225	0.250
4-7_bedrooms	-0.0345	0.013	-2.571	0.010	-0.061	-0.008
4+_bathrooms	0.4767	0.041	11.493	0.000	0.395	0.558
waterfront_1	1.2066	0.063	19.019	0.000	1.082	1.331
grade	0.5096	0.009	58.568	0.000	0.493	0.527

Omnibus:	25.651	**Durbin-Watson:**	1.998
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	31.522
Skew:	-0.001	**Prob(JB):**	1.43e-07
Kurtosis:	3.238	**Cond. No.**	17.2

EVALUATION

Based on the results of my modeling efforts, it has been determined that the predictors that exert the most significant impact on the sale price are, in order of importance: waterfront view, grade of the property, number of bathrooms, and living area.

DEPLOYMENT
DISCOVER UNDERVALUED PROPERTIES WITH PREDICTIVE MODELING SOLUTION

With my predictive modeling tool, you have the ability to input various property predictors and receive an estimate of its value. This estimate, based on the sale price, allows you to evaluate whether the property may be overvalued or undervalued.

1: Enter in the predictor values

```
new_row = pd.DataFrame(columns=used_cols)
new_row = new_row.append({'sqft_living': 1300,
                          'sqft_lot': 2700,
                          'yrs_old': 15,
                          '4-7_bedrooms': 1,
                          '4+_bathrooms': 0,
                          'waterfront_1': 0,
                          'grade': 8},
                         ignore_index=True)
```

2: The model predicts the house price based on your inputs.

```
# prediction needs to be scaled and exponentiated
predicted_price = np.exp(new_row_pred_log) * df["price"].std() + df["price"].mean()

predicted_price = int(predicted_price)
print("The predicted property price is ${:.2f}".format(predicted_price))
```

```
The predicted property price is $752626.00
```

NECESSARY REVISIONS TO ENHANCE MODEL ACCURACY

Revisions to Enhance Model Accuracy

In this presentation, the focus was on demonstrating the impact of various predictors on property prices through inferential modelling.

However, to improve the predictive capabilities of the model, further revisions would be necessary.

This would include incorporating additional predictors and fine-tuning the model, worrying less about multicollinearity, to increase the adjusted R-squared value, thereby enhancing the overall accuracy of the predictive modelling solution.

PRESENTED BY

WARREN MORELLI

warren@momo-mktg.com
GitHub: @Warren-Morelli
LinkedIn:
https://www.linkedin.com/in/warren-morelli/