KING COUNTY HOUSING PROJECT

Edward Cheng

BUSINESS CASE

OVERVIEW

The King County region, located in the US state of Washington, is home to 2.25 million people. An foreign investor has just moved to this County and hired our data science firm. The investor is working closely with a startup construction company that is looking to build houses in this region. The investor would like to understand the following before making any decisions:

- What are the general trends in the housing market?
- What factors drive up the prices of houses?
- Does location have an major effect on prices?
- Is there a way to predict the prices of house for future investing purposes?

PROCESS

Data Cleaning	EDA	Model Building	Final Model
Import data	Examine the general distributions of all the individual predictor variables	Stepwise selection with p-values	Retrieve intercept
Cast columns to the appropriate data types	Question 1	Select features for the model	Retrieve coefficients
Identify and deal with null values appropriately	Question 2	Incorporate OLS modelling	Display model’s formula
Check outliers	Question 3	Feature scaling	Display final R-score
Filter data set	Check linear regression assumptions	Model validation	Analyze results
Deal with categorical variables	Check for multicollinearity		Display final modelling graph
	Apply normalization		

FABRIKAM

```python
# Convert Date column to datetime
df['date'] = pd.to_datetime(df['date'])
```

```python
# Check to see if date is now a numerical data type
print(df.date.dtype)
```

```
datetime64[ns]
```

```python
# For loop that iterates through each entry and calculates the difference between the aforementioned columns
for i in list(df.loc[df['sqft_basement'] == '?'].index.values):
    df.loc[i, 'sqft_basement'] = df.loc[i, 'sqft_living'] - df.loc[i, 'sqft_above']
```

```python
# Confirm now that the 'sqft basement' column is a float data type
df['sqft_basement'] = df['sqft_basement'].astype('float64')
print(df.sqft_basement.dtype)
```

```python
# Observe what percentage of NA values in the total data set are in view
df['view'].isnull().sum() / len(df) * 100
```

```
0.29170718155299347
```

e NA values in the View column constitutes for only 0.3% of the entire dataset, so let's just drop them because it is so insiginficant

```python
df.dropna(subset = ['view'], inplace = True)
df.head()
```

```python
df['waterfront'] = df['waterfront'].fillna(value=0.0)
```

```python
# Check to make sure all the NA values were dropped
df.waterfront.isnull().any()
```

```python
# Check for yr_renovated
df['yr_renovated'].isnull().sum() / len(df) * 100
```

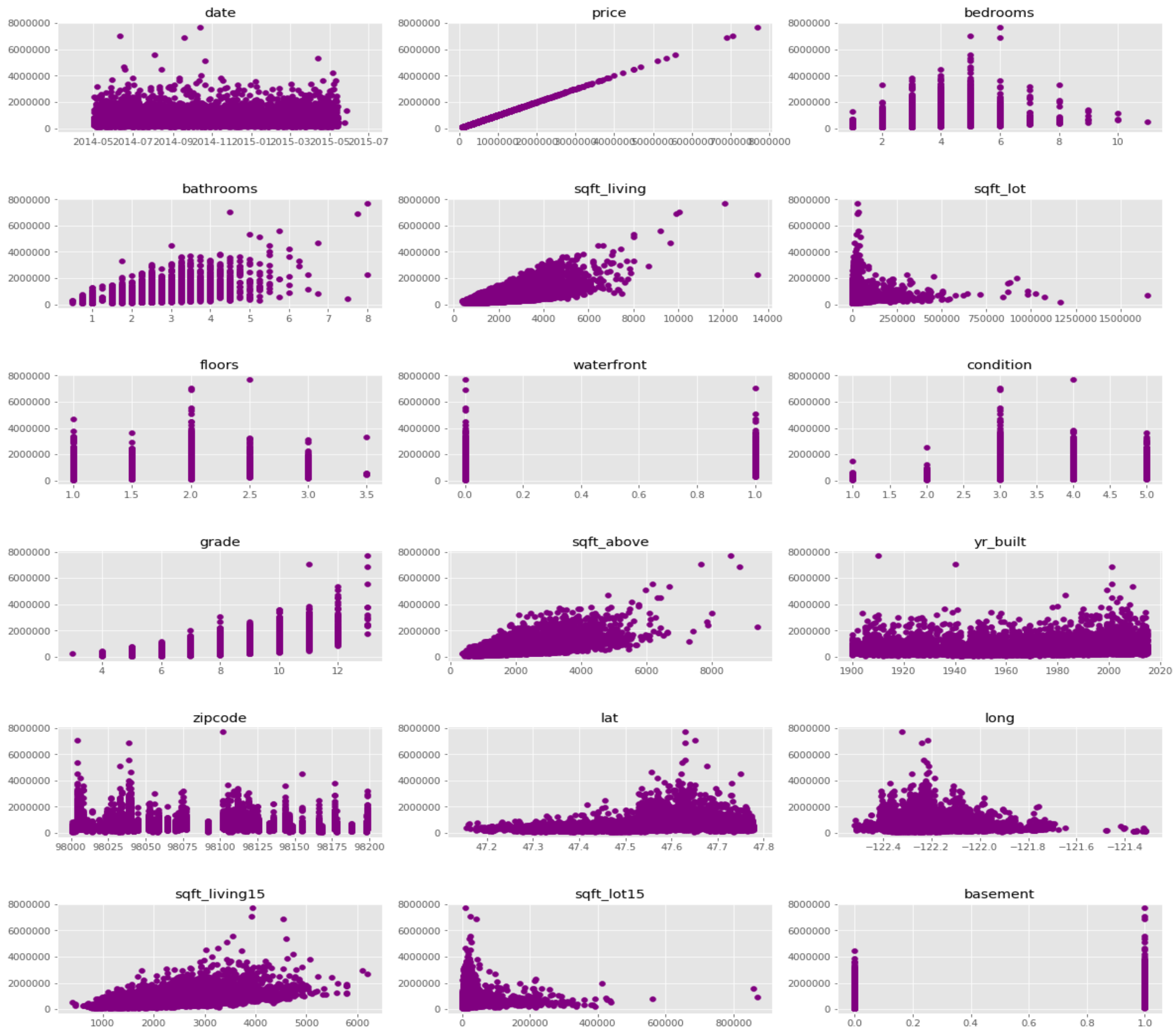
```
17.785827064177578
```

```python
df.yr_renovated.value_counts(normalize = True)
```

DATA CLEANING

Part 1-Data types and null values

- Check index and column data types to identify which data types are not accurate
    - Convert to appropriate data types
- Check for placeholders
- Check for null values using **.isnull().sum()** method
- Use **.value_counts(normalize = True)** to decide whether or not to fill the NA values in with the median or drop them



DATA CLEANING

Part 2- Outliers and Categorical Variables

- Use scatterplot to observe the distributions. The ones that did not follow a more 'linear' pattern are deemed categorical variables
  - **Date**, Bedrooms, Bathrooms, Floors, **Waterfront, Condition**, Grade, Yr_built, **Zipcode, Basement**
- Ordinal vs Nominal variables:
  - The bolded categorical variables are nominal, meaning that they have no order associated with it
    - Ex. A zip code of 95832 is not valued higher than a zip code of 95285

FABRIKAM

```python
# Use Dummy Variables to transform Waterfront, condition, and basement first. Zipcode and Yr_built needs a bit of work t

waterfront_dummies = pd.get_dummies(df['waterfront'], prefix = 'water')
condition_dummies = pd.get_dummies(df['condition'], prefix = 'cond')
basement_dummies = pd.get_dummies(df['basement'], prefix = 'base')

df.drop(["waterfront","condition","basement"], axis=1, inplace = True)

df = pd.concat([df, waterfront_dummies, condition_dummies, basement_dummies], axis = 1)
```

```python
df['zip_categories'] = df['zipcode'].map(return_zipcodes)
zip_dummies = pd.get_dummies(df["zip_categories"], prefix="zip")
df.drop(["zip_categories","zipcode"], axis=1, inplace = True)
df = pd.concat([df, zip_dummies], axis=1)
```

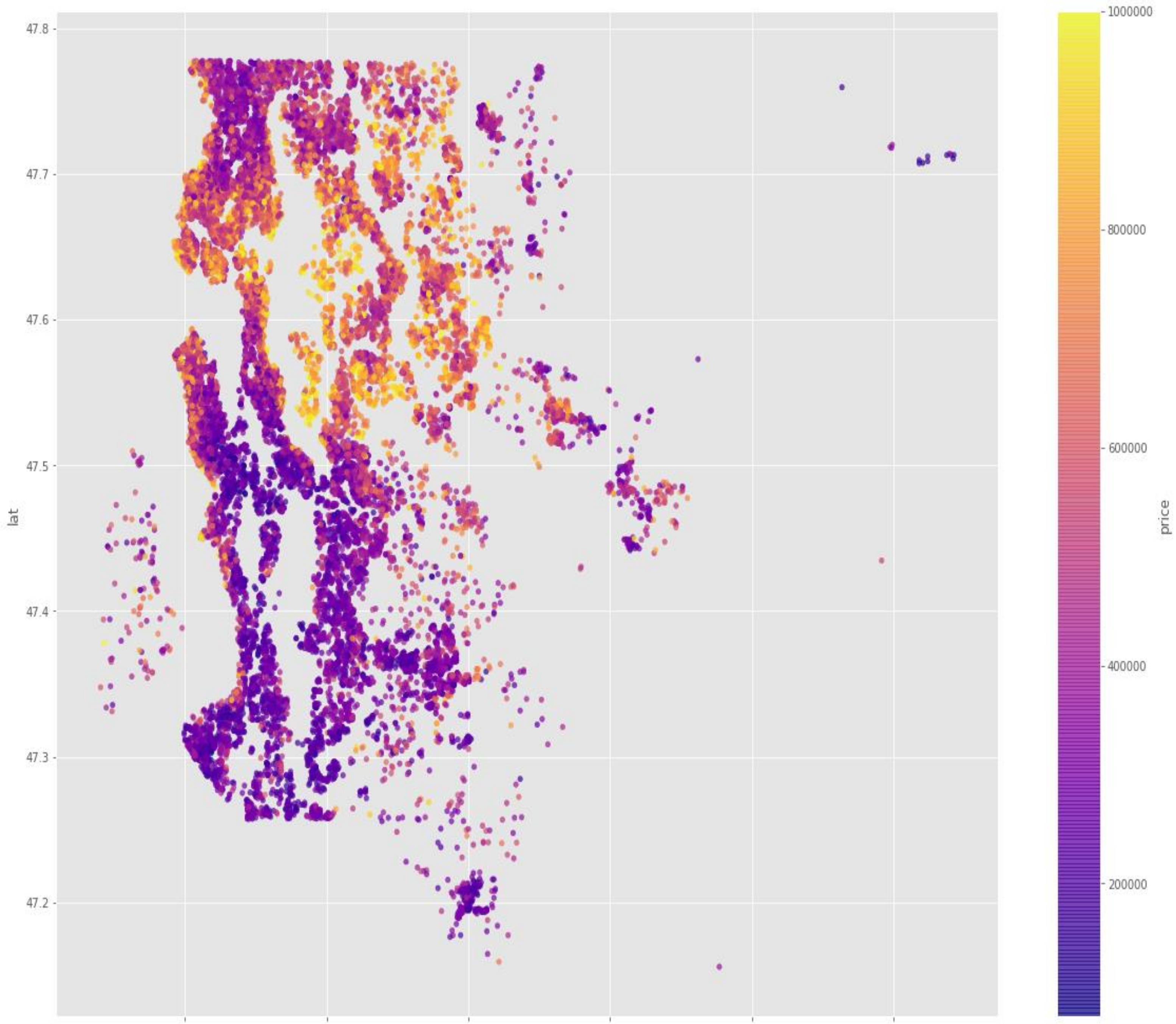
```python
month_dummies = pd.get_dummies(df['date'], prefix="month")
df.drop(["date"], axis = 1, inplace = True)
df = pd.concat([df, month_dummies], axis = 1)
df.head()
```

```python
filtered_drop = df.loc[(df['bedrooms'] > 8) | (df['bathrooms'] > 6) | (df['bathrooms'] < 1) |
                (df['price'] > 1000000) | (df['sqft_living'] > 7000) | (df['sqft_lot'] > 600000) |
                (df['grade'] < 5) | (df['sqft_above'] > 6000) | (df['sqft_lot15'] > 400000)].index
df.drop(filtered_drop, inplace = True)
```

DATA CLEANING

Part 2 Continued…

- Create **dummy variables**, which is the idea of converting each category into a new column, and assign a **1 or 0** to the column
  - Perform this action on the nominal variables
  - Drop the original variable columns after transformation, and concatenate the dummy variable columns to the data frame
  - Zip code was more tricky to work with, as they needed to be organized into area codes from A-I
- Filter data set after cleaning and transforming categorical variables into dummy variables by using **.describe()** to check out values above the **75th percentile** and below the **25th percentile** to see if any value or range of values seemed like outliers



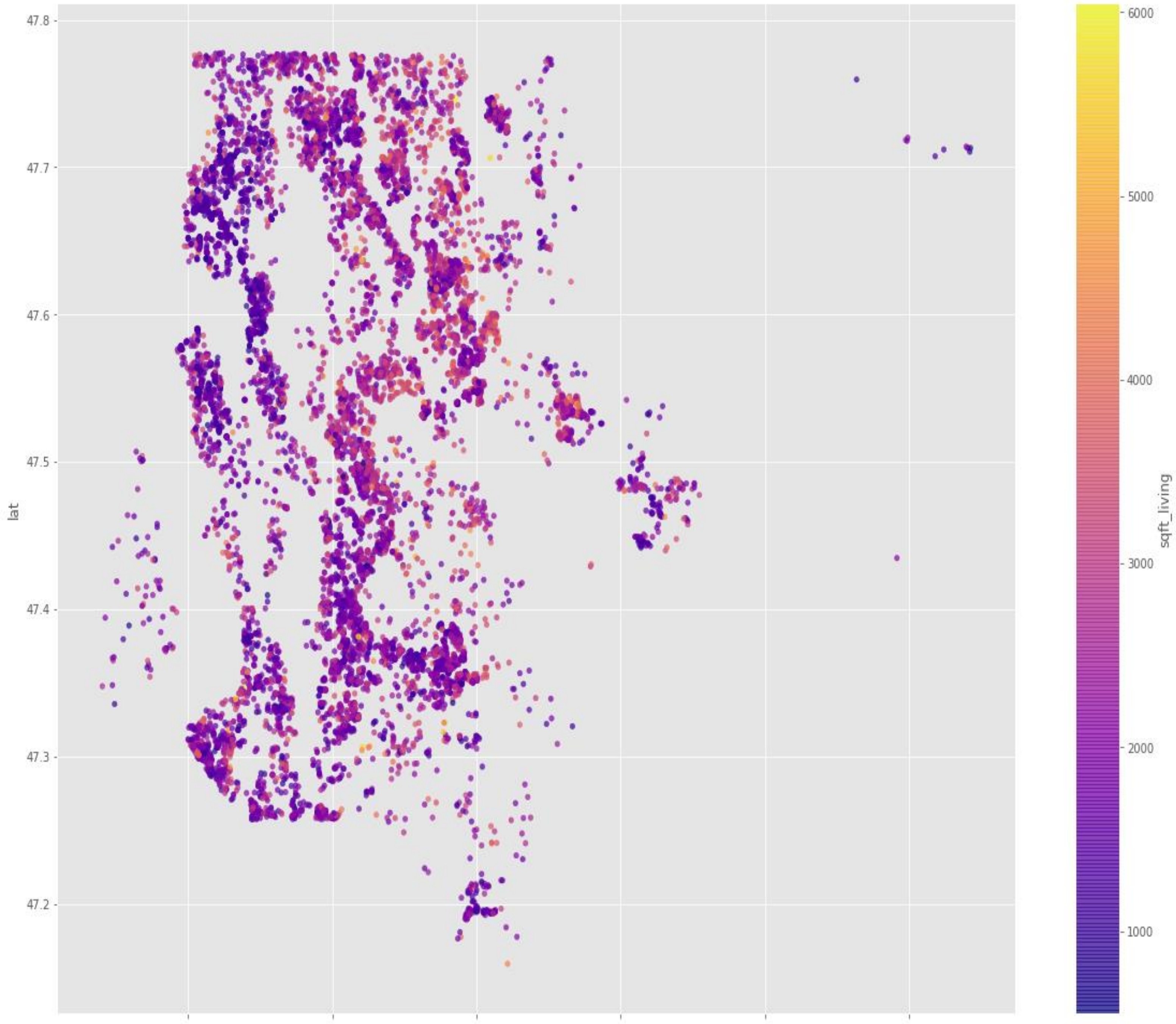
EDA

QUESTION 1

Are housing prices dependent on the location? And if so, do older or newer built houses cost more?

- Higher priced houses tend to be clustered in the North as opposed to the South
- Filter for houses before 1970 and after to check for any trends
    - There are not many expensive houses before 1970 that were in the Northern region
    - Since the last half century, it seems that the more expensive houses generally tend to be cluster farther up North

FABRIKAM



EDA

QUESTION 2

What is the distribution of the sizes of houses? Does having a bigger house equate to having a higher grade?

- Most houses have around the same square footage of land. The bigger houses tend to be spread throughout equally with no obvious clustering
  - The rest of the more average sized houses are found everywhere regardless of the location
- This means that the location itself of where houses are located must be the more important predictor of housing prices
  - We can reasonably infer that the Northern region is the downtown area of the county
- **0.7 correlation** between the size of the house and its grade.
  - That is pretty reasonable, as a house would presumably be rated higher because of its larger housing unit.

FABRIKAM

```
model_2 = ols(formula = 'price ~ bathrooms + bedrooms + floors', data=df).fit()
model_2.summary()
```

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.202
Model:	OLS	**Adj. R-squared:**	0.202
Method:	Least Squares	**F-statistic:**	1687.
Date:	Mon, 06 Jul 2020	**Prob (F-statistic):**	0.00
Time:	19:06:32	**Log-Likelihood:**	-2.6942e+05
No. Observations:	19968	**AIC:**	5.389e+05
Df Residuals:	19964	**BIC:**	5.389e+05
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.474e+05	5603.536	26.309	0.000	1.36e+05	1.58e+05
bathrooms	9.675e+04	2365.232	40.903	0.000	9.21e+04	1.01e+05
bedrooms	2.435e+04	1656.679	14.700	0.000	2.11e+04	2.76e+04
floors	2.88e+04	2693.610	10.692	0.000	2.35e+04	3.41e+04

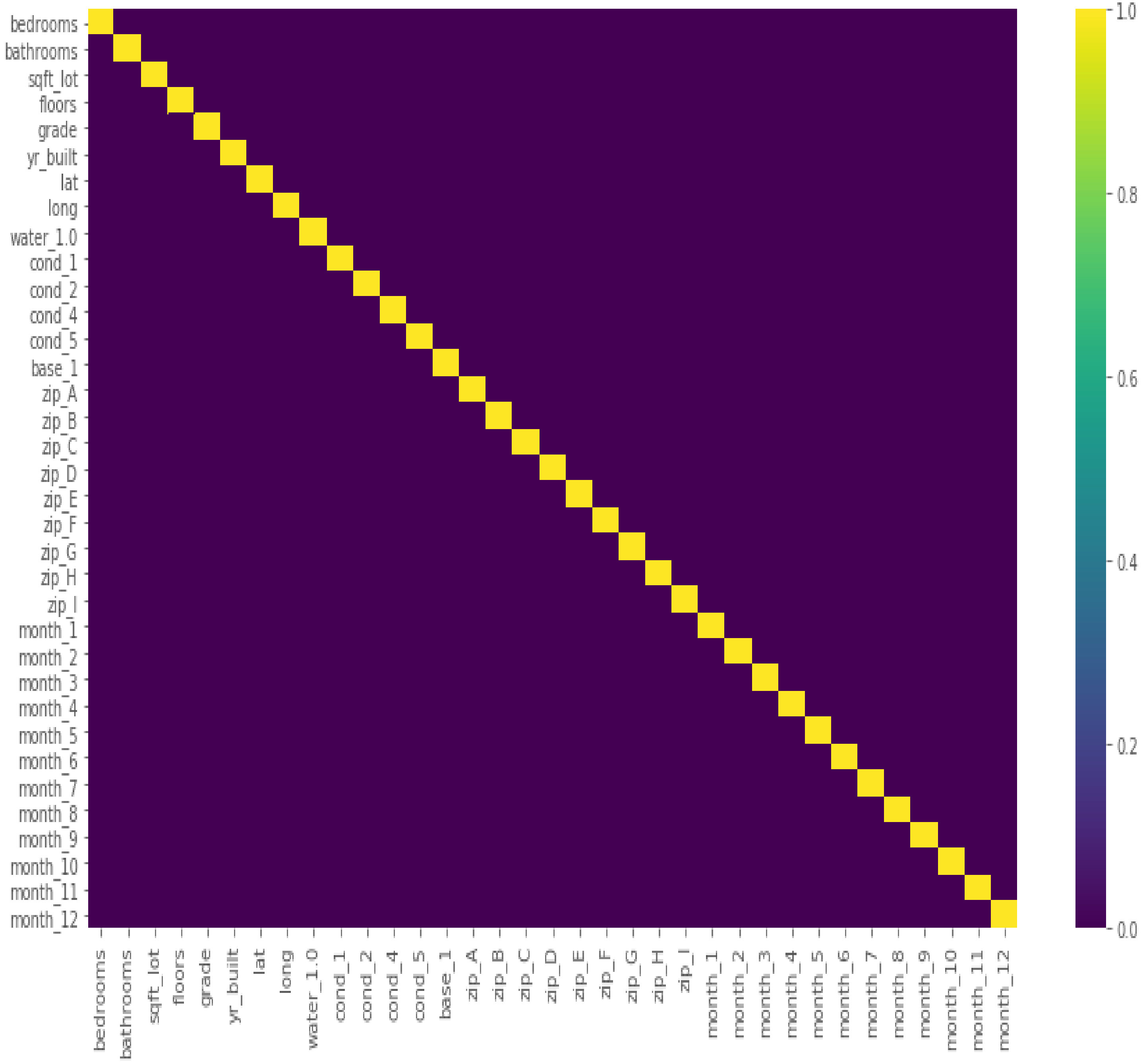
Omnibus:	961.001	**Durbin-Watson:**	1.949
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1090.095
Skew:	0.565	**Prob(JB):**	1.94e-237
Kurtosis:	2.814	**Cond. No.**	20.7

EDA

QUESTION 3

What combination of bathrooms, floors, and/or bedrooms indicates the higher price for houses? Are the findings significant enough?

- Results indicate that a combination of all three variables yields the highest R-squared value
  - Interpretation: A r-squared value of **0.20** means that about 20% of the variance in our target variable ‘price’ is caused by our prediction model with bathrooms, bedrooms, and floors as the predictor variables
  - Generally, a R-squared value of **0.7** or higher is considered acceptable, so these three variables alone will not be enough for our regression model



MULTICOLLINEARITY

Correlation Heatmap

- Check to see if we can eliminate variables that are highly correlated with one another.
    - The assumption in linear regression is that the dependent variable changes based on a change in an independent variable with all other variables held **constant**
- Use a threshold correlation of **>=0.7 & <=1.0** to observe overlapping variable effects
- Use **.stack()** method to output the most highly correlated pair of predictor variables
    - Eliminate the variables **sqft_living**, **sqft_above**, **sqft_living15, sqft_lot15, cond_3**
- Normalize variables afterwards using **log transformations**

FABRIKAM

```python
X_fin = df_features_final
X_with_intercept = sm.add_constant(X_fin)
model = sm.OLS(y,X_with_intercept).fit()
model.summary()
```

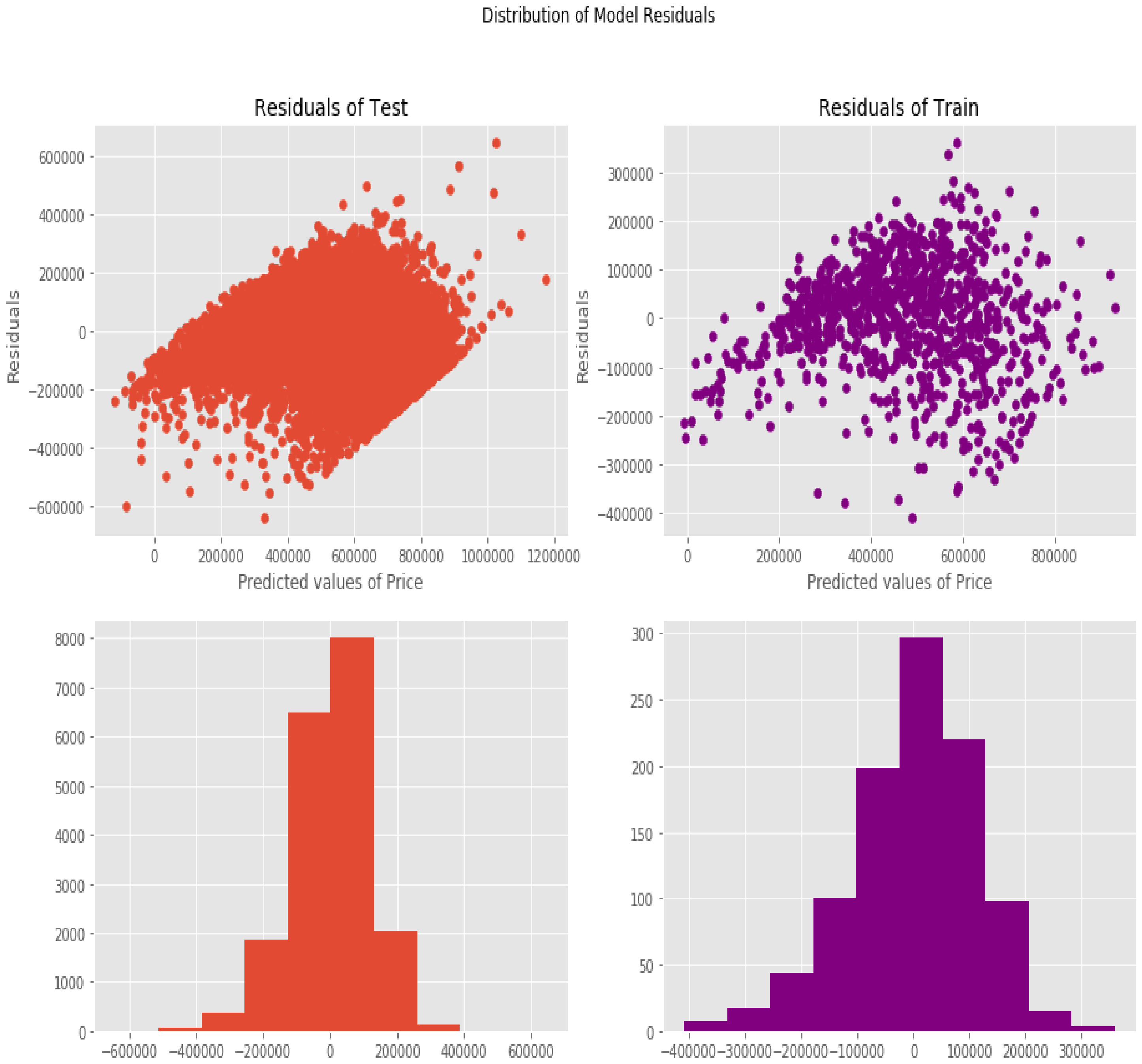
OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.671
Model:	OLS	**Adj. R-squared:**	0.671
Method:	Least Squares	**F-statistic:**	1233.
Date:	Mon, 06 Jul 2020	**Prob (F-statistic):**	0.00
Time:	19:07:09	**Log-Likelihood:**	-2.6057e+05
No. Observations:	19968	**AIC:**	5.212e+05
Df Residuals:	19934	**BIC:**	5.215e+05
Df Model:	33		
Covariance Type:	nonrobust		

FINAL R-SQUARED VALUE

0.671

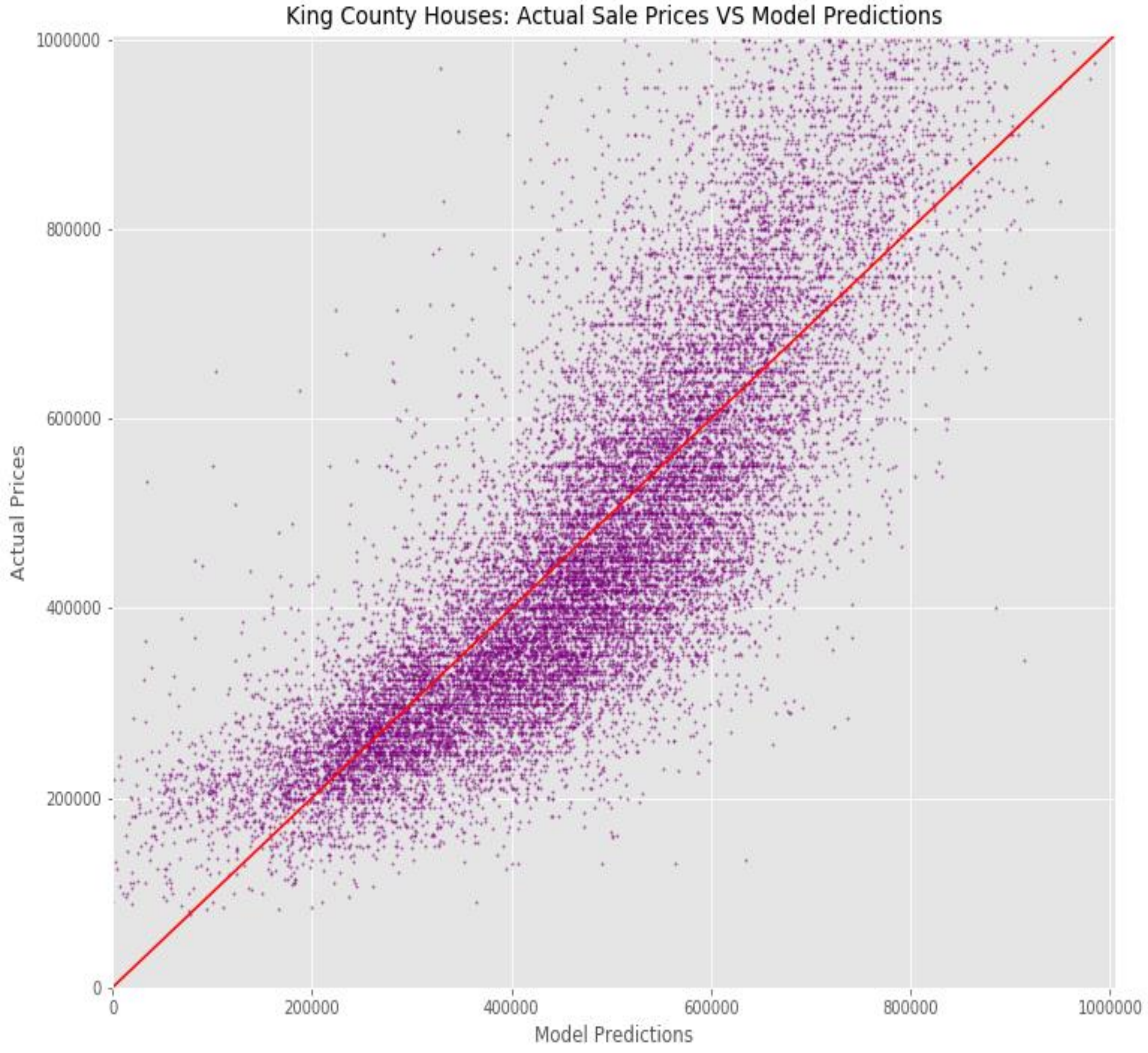
- Perform **stepwise selection** with p-values, then proceeded to scale those chosen features to keep all measurements relative
  - Chosen features: **lat, bedrooms, grade, floors, bathrooms, sqft_lot**
  - Use **MinMaxScaler()** to scale features
- The final model has an r-squared value of **0.671**, meaning that approximately 67% of the variance in our target variable 'price' is caused by our prediction model with the aforementioned variables as the predictor variables



MODEL VALIDATION

Residual Distribution

- Use Train-Test Split validation:
  - Assign **80%** of the data to the training set and **20%** of the data to the testing set
  - Results:
    - Training set MSE: **12690209639.236786**
    - Testing set MSE: **12535566760.226467**
    - Neither underfitted nor overfitted
- Use Cross Validation with **5 Folds**
  - Results:
    - Train-test split MSE: **13055616350.428406**
    - Cross Val 5-Fold MSE: **12780931558.566036**
    - Confirms Train-test split method: model is not underfitted nor overfitted
- The model's residuals appear to be normally distributed so this satisfies the assumption of **homoscedasticity**



FINAL MODEL

Conclusion

$\hat{y}$ = 11022340.24(Lat) + 778260.09(Grade) + 100166.97(Bathrooms) + 78779.72(Floors) + 21272.52(Bedrooms) + 16433.33(Sqft_lot) - 30066101.34

- *Lat*: Can reasonably infer that being farther up North can add a huge amount of value - specifically $*11,022,340.24*
- *Bedrooms*: Having a additional bedroom can bring a modest amount of ***$21,272.52***
- *Grade*: The grading assigned to a housing unit can drive up a house's price massively. As seen in the EDA section, we can confirm this finding as we notice a high correlation between the price of a house and its grade. Grade brings in ***$778,260.09*** in value
- *Floors*: Floors did not seem to have a big impact in our model's r-squared value but it still racks in ***$78,779.72*** in value
- *Bathrooms*: An additional bathrooms brings in a large amount of ***$100,166.97***, and this seems reasonable because most people would be satisfied with having more bathrooms.
- *Sqft_lot*: As seen in the EDA, although the distribution of the sizes of houses were all very evenly spread out with no clear pattern, sqft_lot still brings in a modest value of ***$16,433.33***

THANK YOU FOR YOUR TIME!

Project for Flatiron Online Data Science Bootcamp Module 2
Instructor: Jeff Herman