Analyzing Housing Data

Housing Data

For this project, I decided to look at some simple housing data to improve my EDA skills

The question I am asking is “What variable is the best predictor for price?”

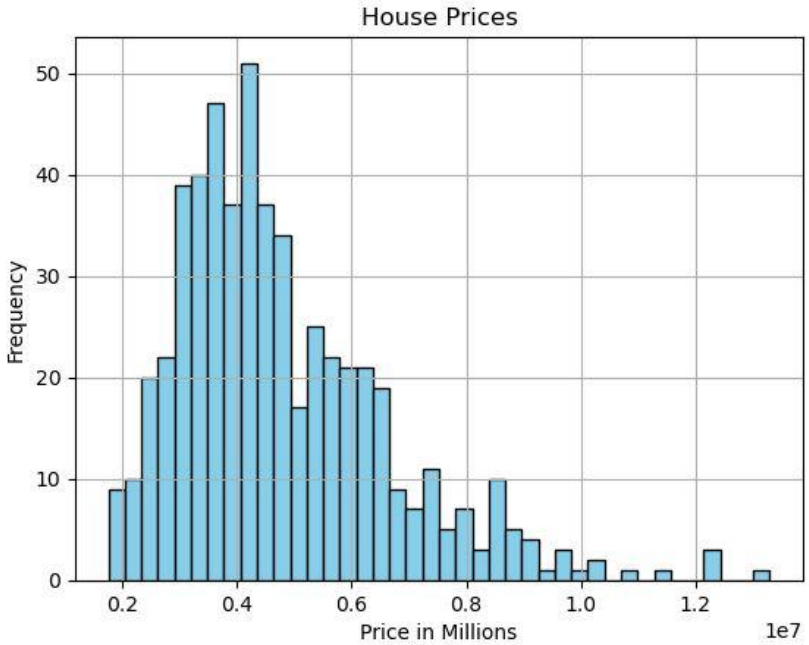
Variables Chosen

1. Price
2. Area
3. Number of Bedrooms
4. Number of Bathrooms
5. Number of Stories

What they mean

1. The price the house sold for
2. Amount of square footage in the house
3. How many bedrooms there are
4. How many bathrooms there are
5. How many stories the house has

House Prices Histogram



There are two outliers just after 1.2 million but I think they should be included in the analysis because they are not extreme.

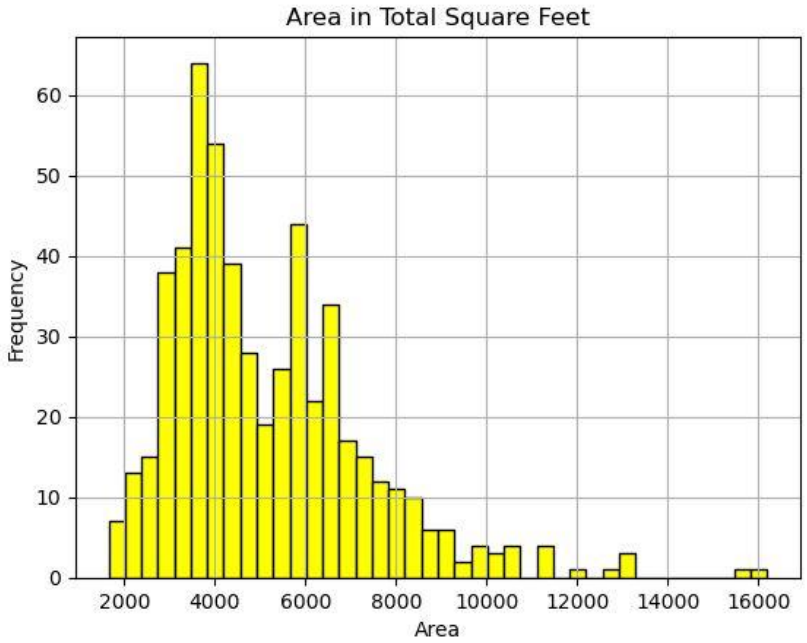
Mean: 476,672

The mode of the variable price is: 350,000

The spread of the variable price is: $100,000 to $1.4 million

The tail trails off to the right.

Area Histogram



There are two outliers around 16000 square feet. I think they should be included because they are not extreme.

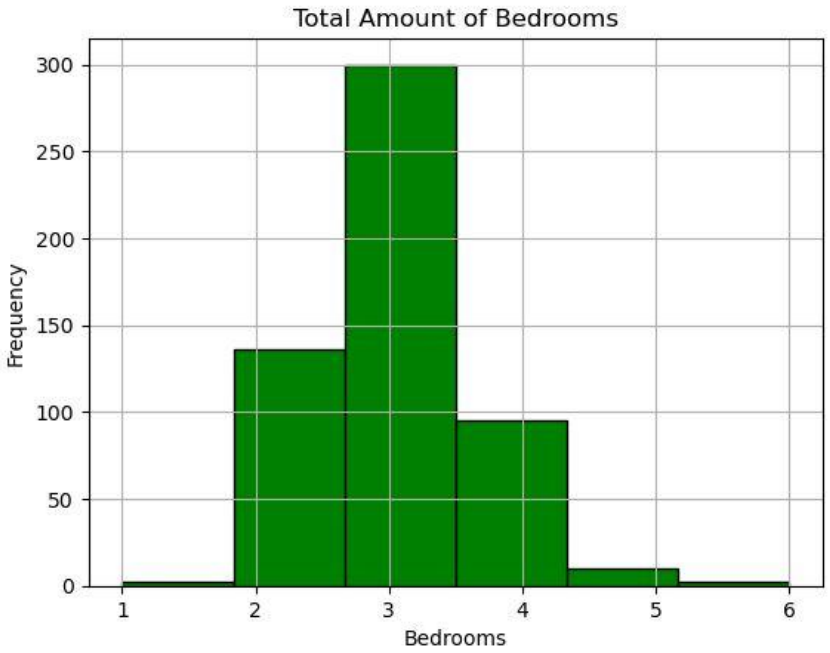
Mean: 5,150 sq ft

Mode: 6,000

Spread: 1,800 to 16,000

The tail trails off to the right

Bedrooms Histogram



No outliers here but you will notice that there are very few 1, 5, and 6 bedrooms houses.

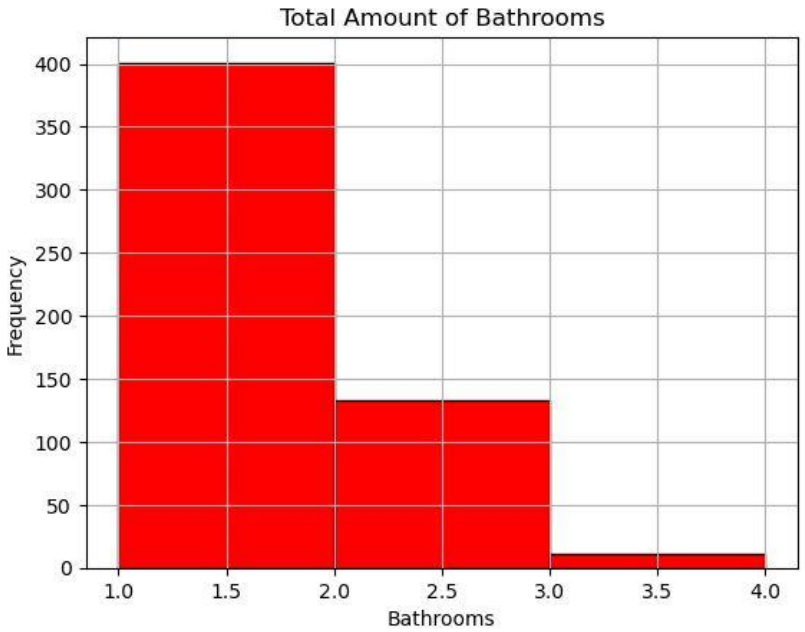
Mean: 3

Mode: 3

Spread: 1 to 6

The tail trails a little to the right

Bathrooms Histogram



No outliers here but you will notice that there are very few houses with 3, 3.5, and 4 bathrooms

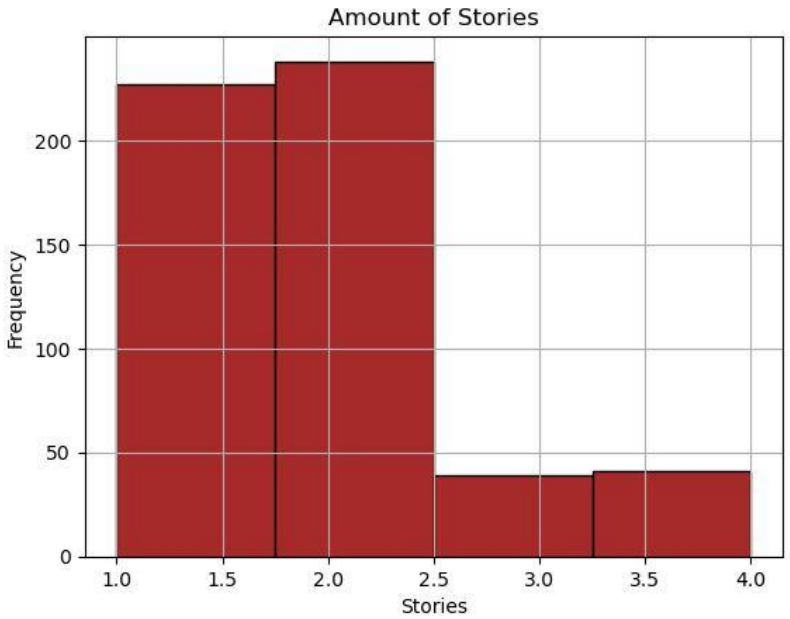
Mean: 1.2

Mode: 1

Spread: 1 to 4

The tail trails off to the right

Bathrooms Histogram



No outliers here but there are fewer 3 and 4 story houses than 1 and 2.

Mean: 1.8

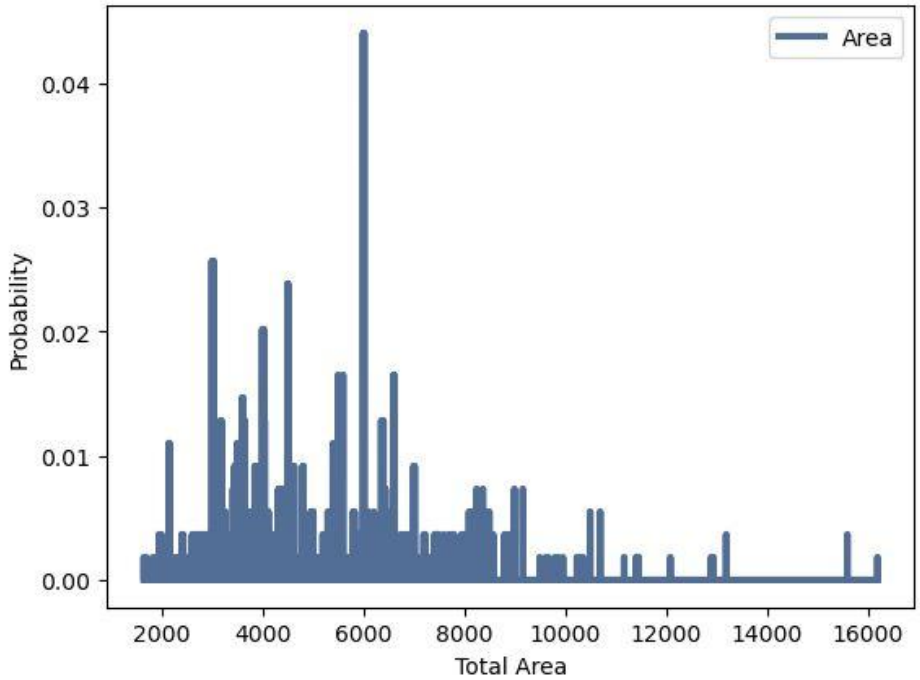
Mode: 2

Spread: 1 to 4

The tail trails off to the right

PMF for Area

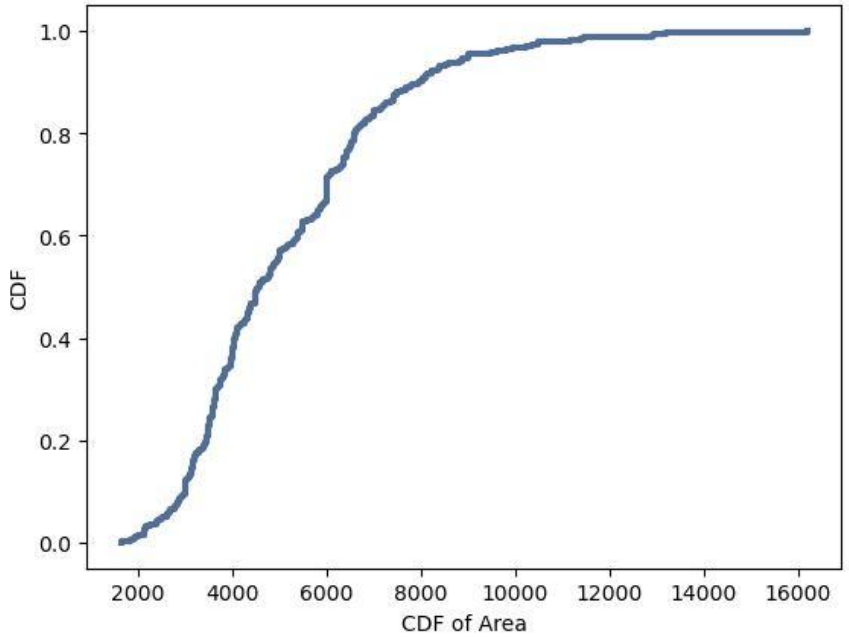
```
pmf = thinkstats2.Pmf(df.area, label="Area")
thinkplot.Pmf(pmf)
thinkplot.Config(xlabel="Total Area", ylabel="Probability")
```



I decided to do a PMF for the area. I ended up with something that looks similar to the histogram. It is different because it shows that 6000 square feet is the most likely to show up, which is also the mode so that makes sense.

CDF for Area

```
cdf = thinkstats2.Cdf(df.area)
thinkplot.Cdf(cdf)
thinkplot.Config(xlabel='CDF of Area', ylabel='CDF')
```

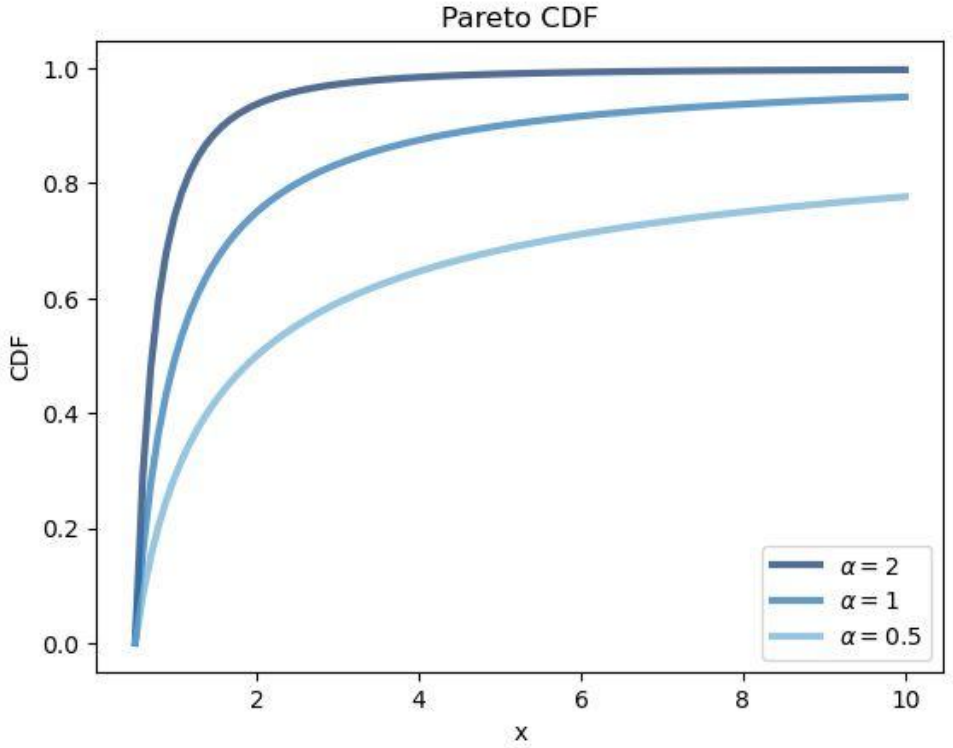


The CDF of the area in each house tells us that the distribution is not normal. It would be more of an S shape if it was normally distributed. This means that most of the houses are 6000 square feet or more.

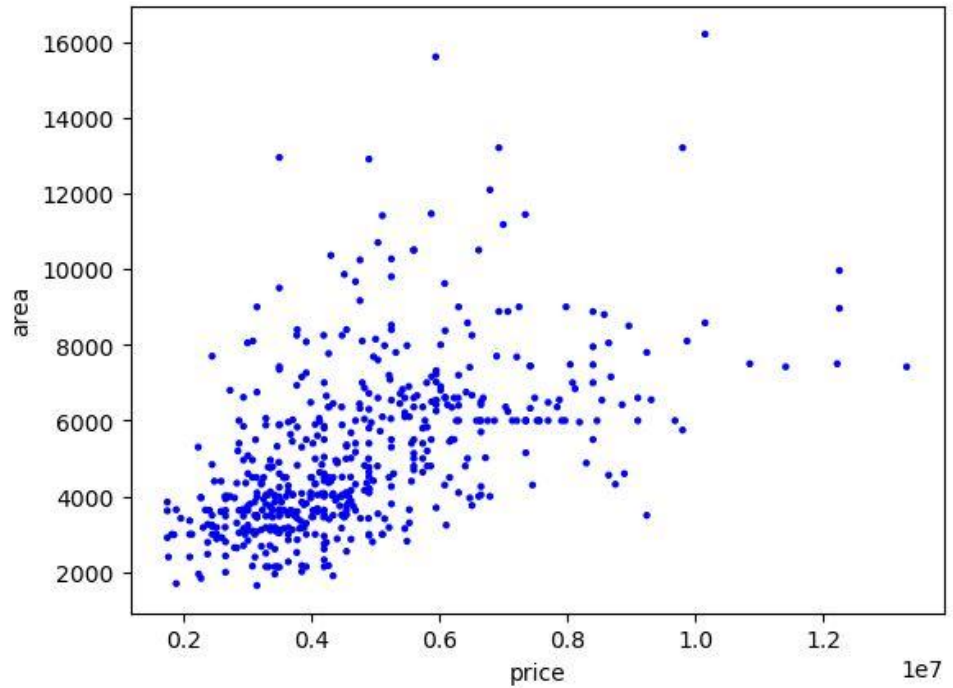
Pareto Distribution

This is a graph of a Pareto Distribution. Notice how the a=1 line is similar to our CDF of the area! This means that roughly 80% of the houses have larger square footage than the rest of the houses.

```
#CDF of the Pareto Distribution
xmin = 0.5

thinkplot.PrePlot(3)
for alpha in [2.0, 1.0, 0.5]:
    xs, ps = thinkstats2.RenderParetoCdf(xmin, alpha, 0, 10.0, n=100)
    thinkplot.Plot(xs, ps, label=r"$\alpha=%g$" % alpha)

thinkplot.Config(title="Pareto CDF", xlabel="x", ylabel="CDF", loc="lower right")
```



```
price = df.price
area = df.area

thinkplot.Scatter(price, area, alpha=1, s=10)
thinkplot.Config(xlabel='price',
                 ylabel='area',
                 legend=False)
```



Scatterplot of Price and Area

This is a scatterplot of the price and area. Notice how there is a concentration at .4 mil and 4000 sq ft. This shows that they are not very strongly correlated so more square footage does not automatically equal a higher price.

```python
def Cov(xs, ys, meanx=None, meany=None):
    xs = np.asarray(xs)
    ys = np.asarray(ys)

    if meanx is None:
        meanx = np.mean(xs)
    if meany is None:
        meany = np.mean(ys)

    cov = np.dot(xs-meanx, ys-meany) / len(xs)
    return cov
```

```python
def Corr(xs, ys):
    xs = np.asarray(xs)
    ys = np.asarray(ys)

    meanx, varx = thinkstats2.MeanVar(xs)
    meany, vary = thinkstats2.MeanVar(ys)

    corr = Cov(xs, ys, meanx, meany) / np.sqrt(varx * vary)
    return corr
```

```python
#Pearson Correlation of area and price
Corr(area, price)
```

```
0.5359973457780802
```

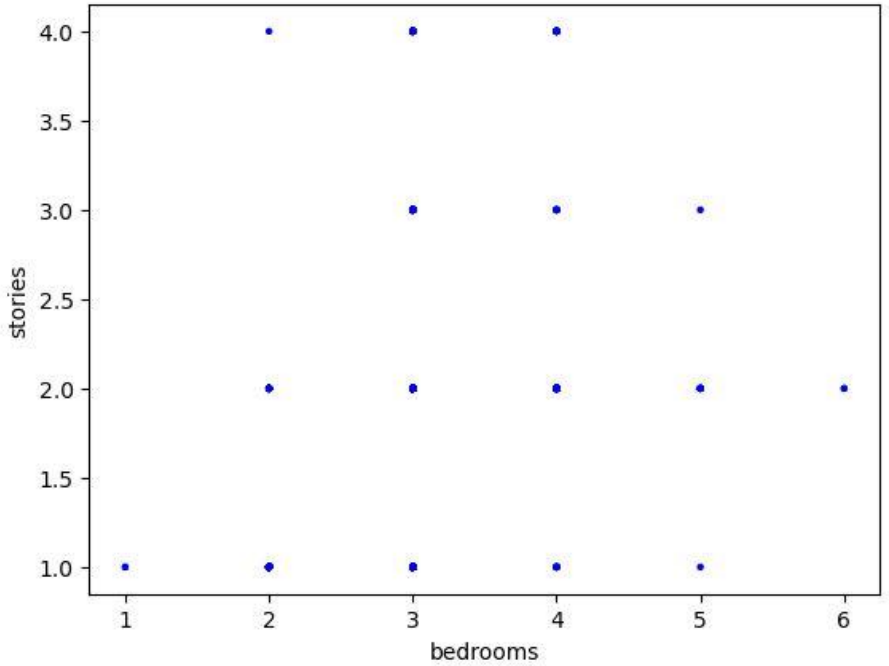
Pearson Correlation Test

The Pearson Correlation being .5 means that they have a positive relation but not a very strong relationship. Looking at the scatterplot, you can see that there are many houses at the same square footage with low and high prices. This lead me to believe that area is not a strong predictor of price and that there is instead something else not in this data that is a stronger predictor.

Scatterplot for Stories and Bedrooms

This is a pretty uninteresting scatterplot as the data is not very granular but it interesting that this shows that the number of stories does not always equal more rooms. The Pearson Correlation test of .4 shows that they have a positive but weak relationship.

```
bedrooms = df.bedrooms
stories = df.stories

thinkplot.Scatter(bedrooms, stories, alpha=1, s=10)
thinkplot.Config(xlabel='bedrooms',
                 ylabel='stories',
                 legend=False)
```




```
#Pearson Correlation for stories and bedrooms
Corr(stories, bedrooms)
```

```
0.40856423753815135
```

Regression Modeling

Here is the start of fitting a regression model to see if area is a good indicator for price. This model shows that each increase of price sees an increase of $428 in price.

```
X = df[['area']]
y = df['price']
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
```

```
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
reg.fit(X_train, y_train)
```

▸ LinearRegression

```
attributes_coefficients = pd.DataFrame(reg.coef_, X.columns, columns=['Coefficient'])
attributes_coefficients
```

	Coefficient
area	428.472666

```
#Each increase of the area sees an increase of about $428 in price.
```

Regression Modeling cont.

Judging the the RMSE being such a high number, this model is not a good fit. It also shows that the area of a house is not a good predictor for the price. There must be other data that would be a better indicator such as location.

```
y_pred = reg.predict(X_test)

comparison = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
comparison
```

	Actual	Predicted
316	4060000	5.072844e+06
77	6650000	5.329928e+06
360	3710000	4.275885e+06
90	6440000	4.687219e+06
493	2800000	4.241608e+06
...	...	...
172	5250000	6.144026e+06
124	5950000	5.340640e+06
388	3500000	4.108781e+06
521	2408000	4.102354e+06
503	2660000	4.258746e+06

137 rows × 2 columns

```
from sklearn import metrics
print('MAE:', metrics.mean_absolute_error(y_test, y_pred))
print('MSE:', metrics.mean_squared_error(y_test, y_pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
MAE: 1401283.7270683737
MSE: 3296594592847.056
RMSE: 1815652.6630517899
```

```
#The RMSE being very large means that this is not the best regression model.
```