King County Housing with Multiple Linear Regression

Authors: Diane Tunnicliffe, Dana Rausch, Matthew Lipman

Notebook 3: Models and Evaluations ¶

This notebook contains linear regression models for our raw, cleaned, and transformed data. We attempted many variations of our model and improved upon them with each iteration to find the best fit for our data. This notebook includes the ten iterations of the model, along with the steps taken to improve them, as well as exploration of necessary assumptions and outputs. The models are evluated sequentially and culminate in a final evaluation and conclusion.

In [5]:
```
# importing the packages we will be using for this project
import pandas as pd
# setting pandas display to avoid scientific notation in my dataframes
pd.options.display.float_format = '{:.2f}'.format
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn

from bs4 import BeautifulSoup
import json
import requests

import folium

import haversine as hs

import statsmodels.api as sm
from statsmodels.formula.api import ols
from statsmodels.stats import diagnostic as diag
from statsmodels.stats.outliers_influence import variance_inflation_factor

from sklearn.metrics import r2_score
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import NearestNeighbors
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error

import scipy.stats as stats

import pylab

%matplotlib inline
```

Model #1

Our first model takes the original raw data and features, within one standard deviation of the mean for price.

In [6]:
```
df = pd.read_csv('./data/all_features_with_logs.csv', index_col=0)
```

In [7]:
```
# define features and target
features = ['sqft_living', 'closest_distance_to_top_school', 'min_dist_park', 'closest_distance_to_great_coffee', 'closest_distance_to_scientology']
target = ['price']

# separate dataframe into feature matrix x and target vector y
X = df[features]
y = df[target]

# now we can instantiate our linear regression estimator and fit our data
lm1 = LinearRegression()
lm1.fit(X, y)

lm1_preds = lm1.predict(X)

print('R^2: ', r2_score(y, lm1_preds))
```

```
R^2:  0.535898617659569
```

In [8]:
```
formula = "price ~ sqft_living+closest_distance_to_top_school+min_dist_park+closest_distance_to_great_coffee+closest_distance_to_scientology"
model = ols(formula= formula, data=df).fit()
model.summary()
```

Out[8]:

OLS Regression Results

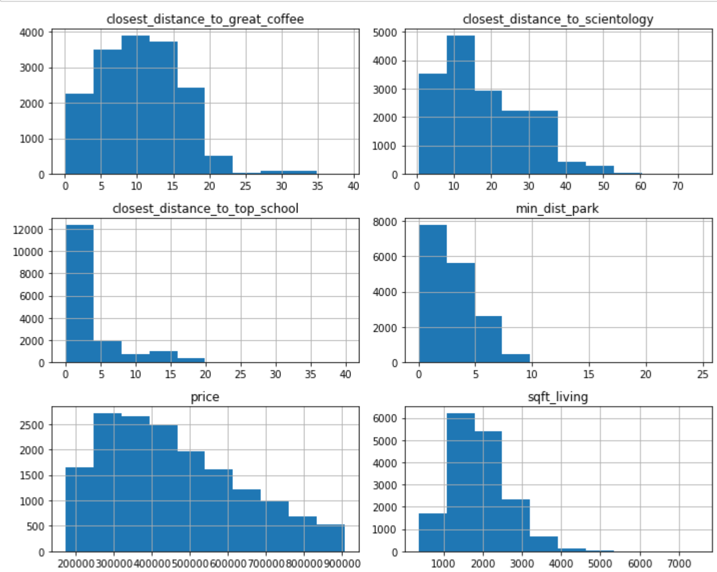
Dep. Variable:	price	**R-squared:**	0.536
Model:	OLS	**Adj. R-squared:**	0.536
Method:	Least Squares	**F-statistic:**	3808.
Date:	Mon, 14 Dec 2020	**Prob (F-statistic):**	0.00
Time:	16:15:17	**Log-Likelihood:**	-2.1650e+05
No. Observations:	16493	**AIC:**	4.330e+05
Df Residuals:	16487	**BIC:**	4.331e+05
Df Model:	5		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	2.716e+05	3896.452	69.717	0.000	2.64e+05	2.79e+05
sqft_living	153.3918	1.374	111.663	0.000	150.699	156.084
closest_distance_to_top_school	-1.006e+04	301.007	-33.405	0.000	-1.06e+04	-9465.225
min_dist_park	-159.3991	468.743	-0.340	0.734	-1078.185	759.387
closest_distance_to_great_coffee	276.8528	183.575	1.508	0.132	-82.973	636.679
closest_distance_to_scientology	-4344.5618	114.862	-37.824	0.000	-4569.704	-4119.419

Omnibus:	365.949	**Durbin-Watson:**	1.993
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	405.658
Skew:	0.341	**Prob(JB):**	8.18e-89
Kurtosis:	3.355	**Cond. No.**	8.45e+03

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 8.45e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

In [9]:
```
# checking the visual distribution of our data with histograms
df[['sqft_living', 'closest_distance_to_great_coffee', 'min_dist_park', 'closest_distance_to_top_school', 'closest_distance_to_scientology', 'price']].hist(figsi
ze=(10,8))
plt.tight_layout();
```

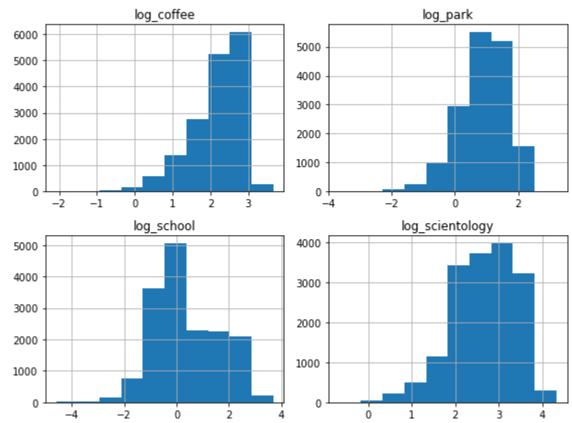


Our distributions for our features were not normal. Please see previous notebook for full investigation of this, analysis of skew and kurtosis, and decision-making regarding transformations.

Model #2

We performed a log-transformation for some of our features to see if this helped to achieve a more normal distribution and improve our model. (For actual process of log-transforming, and visualizations of each feature before and after log-transformation, please see previous notebook titled 'data_wrangling'.)

In [10]:
```
# displaying the visual distribution of our log-transformed data with histograms
df[['log_coffee', 'log_park', 'log_school', 'log_scientology']].hist(figsize=(8,6))
plt.tight_layout();
```



For the full visualizations (sns.distplot) of each feature before and after log-transformation, please see previous notebook ('data_wrangling.ipynb').

In [11]:
```
features = ['sqft_living', 'log_school', 'log_park', 'log_scientology', 'log_coffee']
target = ['price']

X = df[features]
y = df[target]

lm2 = LinearRegression().fit(X, y)

lm2_preds = lm2.predict(X)

print('R^2: ', r2_score(y, lm2_preds))
```

```
R^2:  0.5708370312050253
```

In [12]:
```
formula = "price ~ sqft_living+log_school+log_park+log_scientology+log_coffee"
model = ols(formula= formula, data=df).fit()
```

In [13]:
```
model.summary()
```

Out[13]:

OLS Regression Results

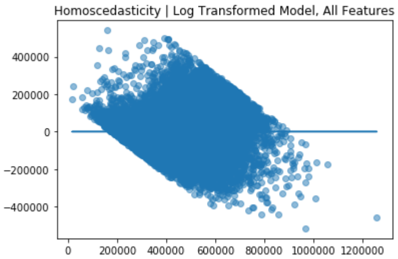
Dep. Variable:	price	**R-squared:**	0.571
Model:	OLS	**Adj. R-squared:**	0.571
Method:	Least Squares	**F-statistic:**	4386.
Date:	Mon, 14 Dec 2020	**Prob (F-statistic):**	0.00
Time:	16:15:19	**Log-Likelihood:**	-2.1585e+05
No. Observations:	16493	**AIC:**	4.317e+05
Df Residuals:	16487	**BIC:**	4.318e+05
Df Model:	5		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	4.24e+05	6263.645	67.686	0.000	4.12e+05	4.36e+05
sqft_living	157.3532	1.315	119.694	0.000	154.776	159.930
log_school	-3.657e+04	958.235	-38.169	0.000	-3.85e+04	-3.47e+04
log_park	-612.2527	1211.889	-0.505	0.613	-2987.686	1763.181
log_scientology	-7.486e+04	1693.247	-44.211	0.000	-7.82e+04	-7.15e+04
log_coffee	-2.526e+04	1385.949	-18.226	0.000	-2.8e+04	-2.25e+04

Omnibus:	342.683	**Durbin-Watson:**	1.987
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	427.218
Skew:	0.283	**Prob(JB):**	1.70e-93
Kurtosis:	3.549	**Cond. No.**	1.46e+04

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.46e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

In [14]:
```
predictors_log = ['sqft_living', 'log_school', 'log_scientology', 'log_coffee', 'log_park']

plt.scatter(model.predict(df[predictors_log]), model.resid, alpha = .5);
plt.plot(model.predict(df[predictors_log]), [0 for i in range(len(df))]);
plt.title('Homoscedasticity | Log Transformed Model, All Features');
```



The variability of price is not equal at all; this model is heteroscedastic. While this iteration increased our R2 score some, we still hoped to achieve a higher one.

Model #3

To attempt to increase our R2 score, we then tried removing certain features to see if the score increased.

In [15]:
```
df.corr()
```

Out[15]:

	price	sqft_living	grade	lat	long	min_dist_park	closest_distance_to_top_school	closest_distance_to_great_coffee	closest_distance_to_scientology	log_school	log_coffee	log_sciento
price	1.00	0.56	0.57	0.45	0.07	0.01	-0.42	-0.20	-0.34	-0.41	-0.17	-
sqft_living	0.56	1.00	0.68	-0.02	0.27	0.01	0.02	-0.13	0.17	0.08	-0.12	
grade	0.57	0.68	1.00	0.05	0.25	0.01	-0.03	-0.14	0.11	0.01	-0.12	
lat	0.45	-0.02	0.05	1.00	-0.13	0.01	-0.68	-0.16	-0.73	-0.63	-0.07	-
long	0.07	0.27	0.25	-0.13	1.00	-0.01	0.01	-0.35	0.63	0.13	-0.39	
min_dist_park	0.01	0.01	0.01	0.01	-0.01	1.00	0.01	0.01	-0.01	0.00	0.01	-
closest_distance_to_top_school	-0.42	0.02	-0.03	-0.68	0.01	0.01	1.00	0.35	0.66	0.86	0.25	
closest_distance_to_great_coffee	-0.20	-0.13	-0.14	-0.16	-0.35	0.01	0.35	1.00	0.14	0.17	0.92	-
closest_distance_to_scientology	-0.34	0.17	0.11	-0.73	0.63	-0.01	0.66	0.14	1.00	0.66	0.03	
log_school	-0.41	0.08	0.01	-0.63	0.13	0.00	0.86	0.17	0.66	1.00	0.12	
log_coffee	-0.17	-0.12	-0.12	-0.07	-0.39	0.01	0.25	0.92	0.03	0.12	1.00	-
log_scientology	-0.33	0.20	0.13	-0.63	0.62	-0.00	0.57	-0.04	0.93	0.63	-0.13	
log_park	0.01	0.02	0.02	0.00	-0.01	0.90	0.01	0.02	-0.00	0.00	0.01	-

Distance to parks seemed to have a relatively low correlation with price, so we experimented with removing that first.

In [16]:
```
features = ['sqft_living', 'log_school', 'log_scientology', 'log_coffee']
target = ['price']
X = df[features]
y = df[target]

lm3 = LinearRegression().fit(X, y)

lm3_preds = lm3.predict(X)

print('R^2: ', r2_score(y, lm3_preds))
```

```
R^2:  0.5708303874090539
```

In [17]:
```
formula = "price ~ sqft_living+log_school+log_scientology+log_coffee"
model = ols(formula= formula, data=df).fit()
```

In [18]:
```
model.summary()
```

Out[18]:

OLS Regression Results

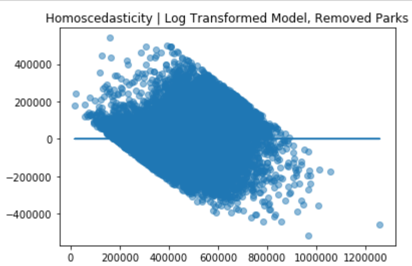
Dep. Variable:	price	**R-squared:**	0.571
Model:	OLS	**Adj. R-squared:**	0.571
Method:	Least Squares	**F-statistic:**	5483.
Date:	Mon, 14 Dec 2020	**Prob (F-statistic):**	0.00
Time:	16:15:19	**Log-Likelihood:**	-2.1585e+05
No. Observations:	16493	**AIC:**	4.317e+05
Df Residuals:	16488	**BIC:**	4.318e+05
Df Model:	4		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	4.235e+05	6183.469	68.482	0.000	4.11e+05	4.36e+05
sqft_living	157.3384	1.314	119.715	0.000	154.762	159.915
log_school	-3.658e+04	958.209	-38.171	0.000	-3.85e+04	-3.47e+04
log_scientology	-7.486e+04	1693.197	-44.211	0.000	-7.82e+04	-7.15e+04
log_coffee	-2.527e+04	1385.806	-18.234	0.000	-2.8e+04	-2.26e+04

Omnibus:	342.576	**Durbin-Watson:**	1.987
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	426.974
Skew:	0.283	**Prob(JB):**	1.92e-93
Kurtosis:	3.548	**Cond. No.**	1.44e+04

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.44e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

In [19]:
```
predictors_3 = ['sqft_living', 'log_school', 'log_coffee', 'log_scientology']

plt.scatter(model.predict(df[predictors_3]), model.resid, alpha = .5);
plt.plot(model.predict(df[predictors_3]), [0 for i in range(len(df))]);
plt.title('Homoscedasticity | Log Transformed Model, Removed Parks');
```



Once again, the variability of price is not equal at all; this model is heteroscedastic. And although we considered removing distance to parks, our R2 score actually dropped a bit as a result.

Model #4

We attempted a new model with only sqare-foot living space and school as features.

In [20]:
```
# trying with only sqft_living and school

features = ['sqft_living', 'log_school']
target = ['price']
X = df[features]
y = df[target]

lm4 = LinearRegression().fit(X, y)

lm4_preds = lm4.predict(X)

print('R^2: ', r2_score(y, lm4_preds))
```

```
R^2:  0.5184159812175783
```

In [21]:
```
formula = "price ~ sqft_living+log_school"
model = ols(formula= formula, data=df).fit()
```

In [22]:
```
model.summary()
```

Out[22]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.518
Model:	OLS	**Adj. R-squared:**	0.518
Method:	Least Squares	**F-statistic:**	8876.
Date:	Mon, 14 Dec 2020	**Prob (F-statistic):**	0.00
Time:	16:15:20	**Log-Likelihood:**	-2.1680e+05
No. Observations:	16493	**AIC:**	4.336e+05
Df Residuals:	16490	**BIC:**	4.336e+05
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.956e+05	2782.391	70.284	0.000	1.9e+05	2.01e+05
sqft_living	149.2004	1.362	109.564	0.000	146.531	151.870
log_school	-6.475e+04	766.641	-84.462	0.000	-6.63e+04	-6.32e+04

Omnibus:	561.519	**Durbin-Watson:**	1.989
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	689.284
Skew:	0.402	**Prob(JB):**	2.11e-150
Kurtosis:	3.598	**Cond. No.**	5.92e+03

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.92e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

Again, the model performs worse upon removal of features.

Model #5

We tried another model with all features, this time using the train_test_split method to train and test our model.

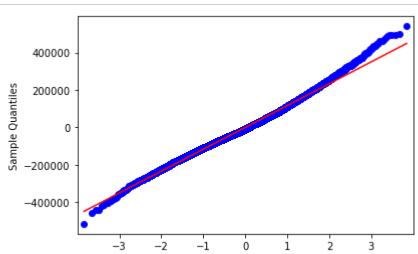
In [23]:
```
features = ['sqft_living', 'log_school', 'log_scientology', 'log_coffee', 'log_park']
target = ['price']
X = df[features]
y = df[target]

# fifth iteration of model: with all and train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, random_state=1)

lm5 = LinearRegression().fit(X_train, y_train)
lm5_preds = lm5.predict(X_test)

print('R^2: ', r2_score(y_test, lm5_preds))
```

```
R^2:  0.5823136171613592
```

In [24]:
```
y_predict = lm5.predict(X_test)

X2 = sm.add_constant(X)

# create an OLS model
model = sm.OLS(y, X2)

# fit the data
est = model.fit()
```

```
/Users/dtunnicliffe/anaconda3/envs/learn-env/lib/python3.6/site-packages/numpy/core/fromnumeric.py:2580: FutureWarning: Method .ptp is deprecated and will be re
moved in a future version. Use numpy.ptp instead.
  return ptp(axis=axis, out=out, **kwargs)
```

In [25]:
```
# check for the normality of the residuals
sm.qqplot(est.resid, line='s')
pylab.show()

# also check that the mean of the residuals is approx. 0.
mean_residuals = sum(est.resid)/ len(est.resid)
print("The mean of the residuals is {:.4}".format(mean_residuals))
```



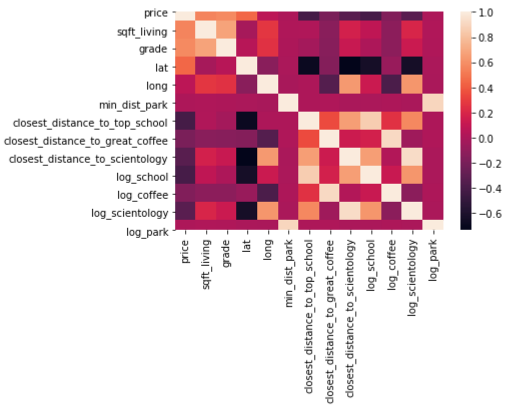

```
The mean of the residuals is 3.426e-10
```

This is the best one so far; the R2 improves when we use all our log-transformed features and train_test_split.

Model #6

We checked for multicolinearity and found that there was multicolinearity between our distance to schools and distance to scientology churches. So we created an interaction column to account for this.

In [26]:
```
sns.heatmap(df.corr());
```



In [27]:
```
df.corr()
```

Out[27]:

	price	sqft_living	grade	lat	long	min_dist_park	closest_distance_to_top_school	closest_distance_to_great_coffee	closest_distance_to_scientology	log_school	log_coffee	log_sciento
price	1.00	0.56	0.57	0.45	0.07	0.01	-0.42	-0.20	-0.34	-0.41	-0.17	-
sqft_living	0.56	1.00	0.68	-0.02	0.27	0.01	0.02	-0.13	0.17	0.08	-0.12	
grade	0.57	0.68	1.00	0.05	0.25	0.01	-0.03	-0.14	0.11	0.01	-0.12	
lat	0.45	-0.02	0.05	1.00	-0.13	0.01	-0.68	-0.16	-0.73	-0.63	-0.07	-
long	0.07	0.27	0.25	-0.13	1.00	-0.01	0.01	-0.35	0.63	0.13	-0.39	
min_dist_park	0.01	0.01	0.01	0.01	-0.01	1.00	0.01	0.01	-0.01	0.00	0.01	-
closest_distance_to_top_school	-0.42	0.02	-0.03	-0.68	0.01	0.01	1.00	0.35	0.66	0.86	0.25	
closest_distance_to_great_coffee	-0.20	-0.13	-0.14	-0.16	-0.35	0.01	0.35	1.00	0.14	0.17	0.92	-
closest_distance_to_scientology	-0.34	0.17	0.11	-0.73	0.63	-0.01	0.66	0.14	1.00	0.66	0.03	
log_school	-0.41	0.08	0.01	-0.63	0.13	0.00	0.86	0.17	0.66	1.00	0.12	
log_coffee	-0.17	-0.12	-0.12	-0.07	-0.39	0.01	0.25	0.92	0.03	0.12	1.00	-
log_scientology	-0.33	0.20	0.13	-0.63	0.62	-0.00	0.57	-0.04	0.93	0.63	-0.13	
log_park	0.01	0.02	0.02	0.00	-0.01	0.90	0.01	0.02	-0.00	0.00	0.01	-

In [28]:
```
# creating an interaction column for school and scientology
# because there is multicolinearity
df['interaction'] = df['log_school'] * df['log_scientology']

features = ['sqft_living', 'log_school', 'log_scientology', 'log_coffee', 'log_park', 'interaction']
target = ['price']

X = df[features]
y = df[target]

# running an iteration of the model with interaction column and using train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, random_state=1)

lm6 = LinearRegression().fit(X_train, y_train)
lm6_preds = lm6.predict(X_test)

print('R^2: ', r2_score(y_test, lm6_preds))
```

```
R^2:  0.5829541835503621
```

In [29]:
```
formula = "price ~ sqft_living+log_school+log_scientology+log_coffee+log_park+interaction"
model = ols(formula= formula, data=df).fit()
model.summary()
```

Out[29]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.571
Model:	OLS	**Adj. R-squared:**	0.571
Method:	Least Squares	**F-statistic:**	3660.
Date:	Mon, 14 Dec 2020	**Prob (F-statistic):**	0.00
Time:	16:15:20	**Log-Likelihood:**	-2.1585e+05
No. Observations:	16493	**AIC:**	4.317e+05
Df Residuals:	16486	**BIC:**	4.318e+05
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	4.225e+05	6273.423	67.355	0.000	4.1e+05	4.35e+05
sqft_living	157.0602	1.317	119.289	0.000	154.479	159.641
log_school	-2.253e+04	3990.855	-5.646	0.000	-3.04e+04	-1.47e+04
log_scientology	-7.487e+04	1692.627	-44.233	0.000	-7.82e+04	-7.16e+04
log_coffee	-2.336e+04	1481.665	-15.764	0.000	-2.63e+04	-2.05e+04
log_park	-614.3072	1211.444	-0.507	0.612	-2988.867	1760.253
interaction	-4698.4477	1296.483	-3.624	0.000	-7239.695	-2157.201

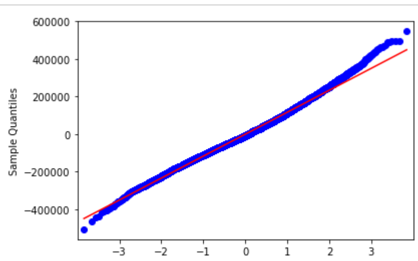
Omnibus:	340.469	**Durbin-Watson:**	1.987
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	419.739
Skew:	0.286	**Prob(JB):**	7.16e-92
Kurtosis:	3.533	**Cond. No.**	1.46e+04

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.46e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

In [30]:
```
y_predict = lm6.predict(X_test)

X2 = sm.add_constant(X)

# create an OLS model
model = sm.OLS(y, X2)

# fit the data
est = model.fit()
```

```
/Users/dtunnicliffe/anaconda3/envs/learn-env/lib/python3.6/site-packages/numpy/core/fromnumeric.py:2580: FutureWarning: Method .ptp is deprecated and will be re
moved in a future version. Use numpy.ptp instead.
  return ptp(axis=axis, out=out, **kwargs)
```

In [31]:
```
# check for the normality of the residuals
sm.qqplot(est.resid, line='s')
pylab.show()

# also check that the mean of the residuals is approx. 0.
mean_residuals = sum(est.resid)/ len(est.resid)
print("The mean of the residuals is {:.4}".format(mean_residuals))
```




```
The mean of the residuals is -4.469e-08
```

This is the best one so far. The model improves when we add an interaction feature.

Model #7

We wanted to include 'grade' as a feature. This is a categorical variable found in the kc_housing dataset. The breakdown for the meaning of each grade designation can be found at https://info.kingcounty.gov/assessor/esales/Glossary.aspx?type=r (https://info.kingcounty.gov/assessor/esales/Glossary.aspx?type=r) under 'Building Grade.'

In [32]:
```
# creating categorical dummy variables for grade
grade_dums = pd.get_dummies(df.grade, prefix='grade', drop_first=True)
```

In [33]:
```
# dropping original grade column
df = df.drop(['grade'], axis=1)
df_with_grade = pd.concat([df, grade_dums], axis=1)
```

In [34]:
```
features = ['sqft_living', 'log_coffee', 'log_park', 'interaction', 'log_school', 'log_scientology', 'grade_4', 'grade_5', 'grade_6', 'grade_7', 'grade_8', 'grad
e_9', 'grade_10', 'grade_11']
target = ['price']
X = df_with_grade[features]
y = df_with_grade[target]

# running an iteration of the model with interaction column and using train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, random_state=1)

lm7 = LinearRegression().fit(X_train, y_train)
lm7_preds = lm7.predict(X_test)

print('R^2: ', r2_score(y_test, lm7_preds))
```

```
R^2:  0.645159498938133
```

In [35]:
```
formula = "price ~ sqft_living+log_coffee+log_park+interaction+log_school+log_scientology+grade_4+grade_5+grade_6+grade_7+grade_8+grade_9+grade_10+grade_11"
model = ols(formula= formula, data=df_with_grade).fit()
model.summary()
```

Out[35]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.637
Model:	OLS	**Adj. R-squared:**	0.637
Method:	Least Squares	**F-statistic:**	2067.
Date:	Mon, 14 Dec 2020	**Prob (F-statistic):**	0.00
Time:	16:15:21	**Log-Likelihood:**	-2.1447e+05
No. Observations:	16493	**AIC:**	4.290e+05
Df Residuals:	16478	**BIC:**	4.291e+05
Df Model:	14		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	7.345e+05	7.64e+04	9.611	0.000	5.85e+05	8.84e+05
sqft_living	98.8114	1.628	60.706	0.000	95.621	102.002
log_coffee	-2.003e+04	1366.438	-14.659	0.000	-2.27e+04	-1.74e+04
log_park	-830.0526	1114.775	-0.745	0.457	-3015.133	1355.027
interaction	-4668.2504	1194.866	-3.907	0.000	-7010.316	-2326.185
log_school	-1.929e+04	3676.138	-5.247	0.000	-2.65e+04	-1.21e+04
log_scientology	-7.909e+04	1563.383	-50.589	0.000	-8.22e+04	-7.6e+04
grade_4	-2.206e+05	8.1e+04	-2.723	0.006	-3.79e+05	-6.18e+04
grade_5	-2.583e+05	7.65e+04	-3.375	0.001	-4.08e+05	-1.08e+05
grade_6	-2.791e+05	7.61e+04	-3.666	0.000	-4.28e+05	-1.3e+05
grade_7	-2.312e+05	7.61e+04	-3.039	0.002	-3.8e+05	-8.21e+04
grade_8	-1.638e+05	7.61e+04	-2.154	0.031	-3.13e+05	-1.48e+04
grade_9	-8.024e+04	7.61e+04	-1.054	0.292	-2.29e+05	6.89e+04
grade_10	-2.103e+04	7.62e+04	-0.276	0.783	-1.7e+05	1.28e+05
grade_11	1.239e+04	7.78e+04	0.159	0.874	-1.4e+05	1.65e+05

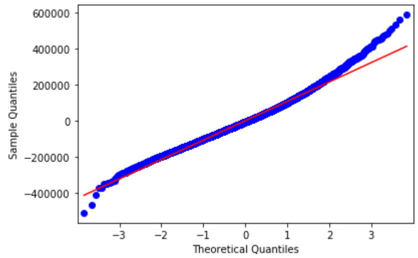
Omnibus:	730.120	**Durbin-Watson:**	1.997
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	993.894
Skew:	0.441	**Prob(JB):**	1.51e-216
Kurtosis:	3.817	**Cond. No.**	5.59e+05

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.59e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

In [36]:
```
y_predict = lm7.predict(X_test)

X2 = sm.add_constant(X)

# create an OLS model
model = sm.OLS(y, X2)

# fit the data
est = model.fit()
```

```
/Users/dtunnicliffe/anaconda3/envs/learn-env/lib/python3.6/site-packages/numpy/core/fromnumeric.py:2580: FutureWarning: Method .ptp is deprecated and will be re
moved in a future version. Use numpy.ptp instead.
  return ptp(axis=axis, out=out, **kwargs)
```

In [37]:
```python
# check for the normality of the residuals
sm.qqplot(est.resid, line='s')
pylab.show()

# also check that the mean of the residuals is approx. 0.
mean_residuals = sum(est.resid)/ len(est.resid)
print("The mean of the residuals is {:.4}".format(mean_residuals))
```




```
The mean of the residuals is -4.565e-08
```

This has once again improved with the addition of the grade column.

Model #8

We then experimented with a quantile transformation of our data, as opposed to a log-transformation.

In [38]:
```python
df = pd.read_csv('./data/all_features_quant_transformed.csv', index_col=0)
df.head()
```

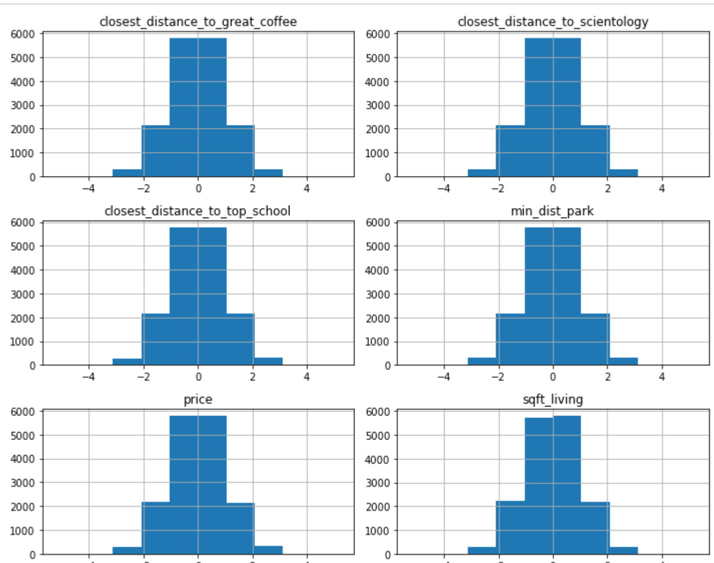
Out[38]:

	price	sqft_living	lat	long	min_dist_park	closest_distance_to_top_school	closest_distance_to_great_coffee	closest_distance_to_scientology	log_school	log_coffee	...	grade_4	grade_5	grade_6	grade_7	grad
0	-1.60	-1.08	47.51	-122.26	-0.31	-1.61	-0.93	-0.24	-1.34	1.60	...	0	0	0	1	
1	0.49	0.94	47.72	-122.32	0.92	-0.50	0.71	-0.40	-0.38	2.70	...	0	0	0	1	
2	-2.54	-2.14	47.74	-122.23	-0.84	0.36	0.09	-0.39	0.69	2.36	...	0	0	1	0	
3	0.78	0.17	47.52	-122.39	-0.08	0.30	0.65	-0.33	0.55	2.67	...	0	0	0	1	
4	0.37	-0.22	47.62	-122.05	0.02	0.08	-0.25	0.37	0.16	2.15	...	0	0	0	0	

5 rows × 22 columns

In [39]:
```python
df.drop(columns=['log_school', 'log_coffee', 'log_scientology', 'log_park'] , axis=1, inplace=True)
```

In [40]:
```python
# checking the visual distribution of our data with histograms
df[['sqft_living', 'closest_distance_to_great_coffee', 'min_dist_park', 'closest_distance_to_top_school', 'closest_distance_to_scientology', 'price']].hist(figsi
ze=(10,8))
plt.tight_layout();
```



In [41]:
```python
features = ['sqft_living', 'closest_distance_to_great_coffee', 'min_dist_park', 'closest_distance_to_top_school', 'closest_distance_to_scientology', 'interactio
n', 'grade_4', 'grade_5', 'grade_6', 'grade_7', 'grade_8', 'grade_9', 'grade_10', 'grade_11']
target = ['price']
X = df[features]
y = df[target]

# running an iteration of the model with quantile transformation and train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, random_state=1)

lm8 = LinearRegression().fit(X_train, y_train)
lm8_preds = lm8.predict(X_test)

print('R^2: ', r2_score(y_test, lm8_preds))
```

```
R^2:  0.6308144610145117
```

In [42]:
```python
formula = "price ~ sqft_living+closest_distance_to_great_coffee+min_dist_park+closest_distance_to_top_school+closest_distance_to_scientology+interaction+grade_4+
grade_5+grade_6+grade_7+grade_8+grade_9+grade_10+grade_11"
model = ols(formula= formula, data=df).fit()
```

In [43]:
```
model.summary()
```

Out[43]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.625
Model:	OLS	**Adj. R-squared:**	0.625
Method:	Least Squares	**F-statistic:**	1961.
Date:	Mon, 14 Dec 2020	**Prob (F-statistic):**	0.00
Time:	16:15:23	**Log-Likelihood:**	-15333.
No. Observations:	16493	**AIC:**	3.070e+04
Df Residuals:	16478	**BIC:**	3.081e+04
Df Model:	14		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.4887	0.434	3.430	0.001	0.638	2.339
sqft_living	0.4005	0.007	60.597	0.000	0.388	0.413
closest_distance_to_great_coffee	-0.0351	0.006	-6.328	0.000	-0.046	-0.024
min_dist_park	-0.0023	0.005	-0.474	0.636	-0.012	0.007
closest_distance_to_top_school	-0.2366	0.006	-37.579	0.000	-0.249	-0.224
closest_distance_to_scientology	-0.3240	0.006	-51.764	0.000	-0.336	-0.312
interaction	-0.0028	0.005	-0.559	0.576	-0.013	0.007
grade_4	-1.3811	0.462	-2.988	0.003	-2.287	-0.475
grade_5	-1.8401	0.437	-4.215	0.000	-2.696	-0.984
grade_6	-1.9693	0.434	-4.535	0.000	-2.821	-1.118
grade_7	-1.6686	0.434	-3.845	0.000	-2.519	-0.818
grade_8	-1.2934	0.434	-2.980	0.003	-2.144	-0.443
grade_9	-0.8742	0.434	-2.013	0.044	-1.726	-0.023
grade_10	-0.5436	0.435	-1.250	0.211	-1.396	0.309
grade_11	-0.2912	0.444	-0.655	0.512	-1.162	0.580

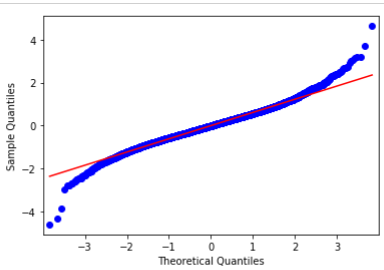
Omnibus:	696.435	**Durbin-Watson:**	2.004
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	2235.973
Skew:	0.085	**Prob(JB):**	0.00
Kurtosis:	4.796	**Cond. No.**	430.

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [44]:
```
y_predict = lm8.predict(X_test)

X2 = sm.add_constant(X)

# create an OLS model
model = sm.OLS(y, X2)

# fit the data
est = model.fit()
```

```
/Users/dtunnicliffe/anaconda3/envs/learn-env/lib/python3.6/site-packages/numpy/core/fromnumeric.py:2580: FutureWarning: Method .ptp is deprecated and will be re
moved in a future version. Use numpy.ptp instead.
  return ptp(axis=axis, out=out, **kwargs)
```

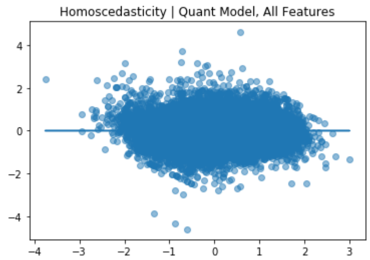
In [45]:
```
# check for the normality of the residuals
sm.qqplot(est.resid, line='s')
pylab.show()

# also check that the mean of the residuals is approx. 0.
mean_residuals = sum(est.resid)/ len(est.resid)
print("The mean of the residuals is {:.4}".format(mean_residuals))
```




```
The mean of the residuals is -1.585e-15
```

Our residuals are relatively normal.

```
In [46]: f = 'price ~ sqft_living+closest_distance_to_great_coffee+min_dist_park+closest_distance_to_top_school+closest_distance_to_scientology+interaction++grade_4+grade
_5+grade_6+grade_7+grade_8+grade_9+grade_10+grade_11'

model = ols(formula = f, data = df).fit()
model.summary()
predictors_quant = ['sqft_living', 'closest_distance_to_great_coffee', 'min_dist_park', 'closest_distance_to_top_school', 'closest_distance_to_scientology', 'int
eraction', 'grade_4', 'grade_5', 'grade_6', 'grade_7', 'grade_8', 'grade_9', 'grade_10', 'grade_11']

plt.scatter(model.predict(df[predictors_quant]), model.resid, alpha = .5);
plt.plot(model.predict(df[predictors_quant]), [0 for i in range(len(df))]);
plt.title('Homoscedasticity | Quant Model, All Features');
```



Our qq-plots, homoscedasticity, and R-squared value continue to improve with each iteration.

Model #9

We then experimented with a target we created, Price Per Square-Foot. While this target unfortunately decreased our R2 significantly, we were able to use this new variable we'd created as a new measurement by which to remove outliers and narrow our data further. Our last model retains our original price target, but uses data narrowed to 1.5 standard deviations from the mean of price per square foot. (For this entire process, please see previous notebook, 'data_wrangling'.) At this point, we also updated our list of parks to eliminate forests and trail heads, and only include actual parks, to make for a more accurate "distance to closest park" measurement.

```
In [47]: df = pd.read_csv('./data/all_features_ppsqft_quant.csv', index_col=0)
df.head()
```

Out[47]:

	price	sqft_living	lat	long	price_per_sqft	min_dist_park	closest_distance_to_top_school	closest_distance_to_great_coffee	closest_distance_to_scientology	interaction	...	quant_interaction	grade_5	grade_
0	221900.00	1180	47.51	-122.26	188.05	2.04	0.26	4.39	12.71	3.33	...	-1.11	0	
1	538000.00	2570	47.72	-122.32	209.34	5.67	0.68	14.81	10.80	7.37	...	-0.50	0	
2	180000.00	770	47.74	-122.23	233.77	1.34	2.00	10.63	10.84	21.71	...	0.08	0	
3	604000.00	1960	47.52	-122.39	308.16	2.45	1.73	14.48	11.55	19.97	...	0.05	0	
4	510000.00	1680	47.62	-122.05	303.57	3.72	1.18	8.55	21.18	24.98	...	0.16	0	

5 rows × 27 columns

```
In [48]: features = ['quant_sqft_living','quant_coffee', 'quant_parks', 'quant_schools', 'quant_scientology', 'grade_5', 'grade_6', 'grade_7', 'grade_8', 'grade_9', 'grad
e_10', 'grade_11', 'grade_12', 'grade_13', 'quant_interaction']
target = ['quant_price']
X = df[features]
y = df[target]

# running an iteration of the model using train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, random_state=1)

lm9 = LinearRegression().fit(X_train, y_train)
lm9_preds = lm9.predict(X_test)

print('R^2: ', r2_score(y_test, lm9_preds))
```

```
R^2:  0.7559870492262424
```

In [49]:
```
formula = "quant_price ~ quant_sqft_living+quant_coffee+quant_parks+quant_schools+quant_scientology+quant_interaction+grade_5+grade_6+grade_7+grade_8+grade_9+grade_10+grade_11+grade_12+grade_13"
model = ols(formula= formula, data=df).fit()
model.summary()
```

Out[49]:

OLS Regression Results

Dep. Variable:	quant_price	**R-squared:**	0.761
Model:	OLS	**Adj. R-squared:**	0.761
Method:	Least Squares	**F-statistic:**	3711.
Date:	Mon, 14 Dec 2020	**Prob (F-statistic):**	0.00
Time:	16:15:24	**Log-Likelihood:**	-12314.
No. Observations:	17495	**AIC:**	2.466e+04
Df Residuals:	17479	**BIC:**	2.479e+04
Df Model:	15		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-0.7602	0.123	-6.167	0.000	-1.002	-0.519
quant_sqft_living	0.4987	0.006	89.561	0.000	0.488	0.510
quant_coffee	-0.0269	0.004	-6.792	0.000	-0.035	-0.019
quant_parks	-0.0059	0.004	-1.595	0.111	-0.013	0.001
quant_schools	-0.0690	0.021	-3.229	0.001	-0.111	-0.027
quant_scientology	-0.1565	0.014	-11.053	0.000	-0.184	-0.129
quant_interaction	-0.2132	0.031	-6.879	0.000	-0.274	-0.152
grade_5	0.1626	0.128	1.274	0.203	-0.088	0.413
grade_6	0.3070	0.123	2.492	0.013	0.066	0.549
grade_7	0.5833	0.123	4.736	0.000	0.342	0.825
grade_8	0.8820	0.124	7.131	0.000	0.640	1.124
grade_9	1.1951	0.125	9.596	0.000	0.951	1.439
grade_10	1.4316	0.126	11.387	0.000	1.185	1.678
grade_11	1.7193	0.129	13.377	0.000	1.467	1.971
grade_12	2.0848	0.144	14.463	0.000	1.802	2.367
grade_13	2.3285	0.236	9.847	0.000	1.865	2.792

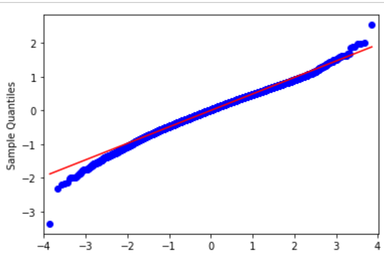
Omnibus:	391.796	**Durbin-Watson:**	1.997
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	511.788
Skew:	-0.283	**Prob(JB):**	7.35e-112
Kurtosis:	3.617	**Cond. No.**	175.

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [50]:
```
y_predict = lm9.predict(X_test)

X2 = sm.add_constant(X)

# create an OLS model
model = sm.OLS(y, X2)

# fit the data
est = model.fit()
```

```
/Users/dtunnicliffe/anaconda3/envs/learn-env/lib/python3.6/site-packages/numpy/core/fromnumeric.py:2580: FutureWarning: Method .ptp is deprecated and will be removed in a future version. Use numpy.ptp instead.
  return ptp(axis=axis, out=out, **kwargs)
```

In [51]:
```
# check for the normality of the residuals
sm.qqplot(est.resid, line='s')
pylab.show()

# also check that the mean of the residuals is approx. 0.
mean_residuals = sum(est.resid)/ len(est.resid)
print("The mean of the residuals is {:.4}".format(mean_residuals))
```




```
The mean of the residuals is -1.626e-15
```

Our residuals are relatively normal.

Recursive Feature Elimination (RFE)

```python
In [53]: # def lin_reg(X, y):
         #    """Recursive feature elimination (RFE) function"""
         #     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25)
         #     linreg = LinearRegression()
         #     linreg.fit(X_train,y_train)
         #     y_hat = linreg.predict(X_test)
         #     y_hat_train = linreg.predict(X_train)
         #     print('R_squared:', linreg.score(X, y))
         #     #Display errors
         #     print('Mean Absolute Error:', mean_absolute_error(y_test, y_hat))
         #     print('Root Mean Squared Error test:', np.sqrt(mean_squared_error(y_test, y_hat)))
         #     print('Root Mean Squared Error train:', np.sqrt(mean_squared_error(y_train, y_hat_train)))
         #     #Compare predicted and actual values
         #     print('Mean Predicted Selling Price:', y_hat.mean())
         #     print('Mean Selling Price:', y_test.mean())
         #     return linreg
```

```python
In [54]: # lin_reg(X,y)
```

```python
In [55]: #RFE to check for insignificant features
         # from sklearn.svm import SVR
         # from sklearn.feature_selection import RFE

         # estimator = SVR(kernel="linear")

         # selector = RFE(estimator, step=1)
         # selector = selector.fit(X, y)

         # #Take a look at the R2 with only the most valuable features
         # X_RFE = X[X.columns[selector.support_]]
         # lin_reg(X_RFE, y)
```

Model #10

We then took our previous model and removed parks as a feature altogether, since further analysis showed that this was not helping our R2 score. For the entire investigation into each feature's impact on the model, please see the notebook titled 'Iterating Through Final Model."

```python
In [56]: features = ['quant_sqft_living','quant_coffee', 'quant_schools', 'quant_scientology', 'grade_5', 'grade_6', 'grade_7', 'grade_8', 'grade_9', 'grade_10', 'grade_1
         1', 'grade_12', 'grade_13', 'quant_interaction']
         target = ['quant_price']
         X = df[features]
         y = df[target]

         # running an iteration of the model using train_test_split
         X_train, X_test, y_train, y_test = train_test_split(X,y, random_state=1)

         lm10 = LinearRegression().fit(X_train, y_train)
         lm10_preds = lm10.predict(X_test)

         print('R^2: ', r2_score(y_test, lm10_preds))
```

```
R^2:  0.7559686827061596
```

In [57]:
```
formula = "quant_price ~ quant_sqft_living+quant_coffee+quant_schools+quant_scientology+quant_interaction+grade_5+grade_6+grade_7+grade_8+grade_9+grade_10+grade_
11+grade_12+grade_13"
model = ols(formula= formula, data=df).fit()
model.summary()
```

Out[57]:

OLS Regression Results

Dep. Variable:	quant_price	**R-squared:**	0.761
Model:	OLS	**Adj. R-squared:**	0.761
Method:	Least Squares	**F-statistic:**	3975.
Date:	Mon, 14 Dec 2020	**Prob (F-statistic):**	0.00
Time:	16:16:10	**Log-Likelihood:**	-12316.
No. Observations:	17495	**AIC:**	2.466e+04
Df Residuals:	17480	**BIC:**	2.478e+04
Df Model:	14		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-0.7595	0.123	-6.162	0.000	-1.001	-0.518
quant_sqft_living	0.4986	0.006	89.550	0.000	0.488	0.510
quant_coffee	-0.0268	0.004	-6.779	0.000	-0.035	-0.019
quant_schools	-0.0690	0.021	-3.229	0.001	-0.111	-0.027
quant_scientology	-0.1564	0.014	-11.045	0.000	-0.184	-0.129
quant_interaction	-0.2133	0.031	-6.882	0.000	-0.274	-0.153
grade_5	0.1622	0.128	1.271	0.204	-0.088	0.412
grade_6	0.3062	0.123	2.486	0.013	0.065	0.548
grade_7	0.5827	0.123	4.730	0.000	0.341	0.824
grade_8	0.8813	0.124	7.125	0.000	0.639	1.124
grade_9	1.1946	0.125	9.592	0.000	0.951	1.439
grade_10	1.4313	0.126	11.385	0.000	1.185	1.678
grade_11	1.7186	0.129	13.371	0.000	1.467	1.971
grade_12	2.0842	0.144	14.458	0.000	1.802	2.367
grade_13	2.3268	0.236	9.839	0.000	1.863	2.790

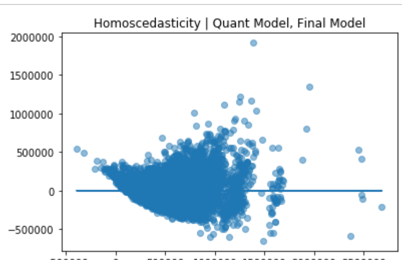
Omnibus:	391.327	**Durbin-Watson:**	1.997
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	510.627
Skew:	-0.283	**Prob(JB):**	1.31e-111
Kurtosis:	3.616	**Cond. No.**	175.

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

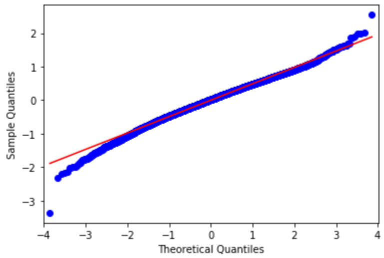
In [58]:
```
y_predict = lm10.predict(X_test)

X2 = sm.add_constant(X)

# create an OLS model
model = sm.OLS(y, X2)

# fit the data
est = model.fit()
```

```
/Users/dtunnicliffe/anaconda3/envs/learn-env/lib/python3.6/site-packages/numpy/core/fromnumeric.py:2580: FutureWarning: Method .ptp is deprecated and will be re
moved in a future version. Use numpy.ptp instead.
  return ptp(axis=axis, out=out, **kwargs)
```

In [59]:
```
f = 'price ~ quant_sqft_living+quant_coffee+quant_schools+quant_scientology+quant_interaction+grade_5+grade_6+grade_7+grade_8+grade_9+grade_10+grade_11+grade_12+
grade_13'
model = ols(formula = f, data = df).fit()

predictors_quant = ['quant_sqft_living','quant_coffee', 'quant_schools', 'quant_scientology', 'grade_5', 'grade_6', 'grade_7', 'grade_8', 'grade_9', 'grade_10',
'grade_11', 'grade_12', 'grade_13', 'quant_interaction']

plt.scatter(model.predict(df[predictors_quant]), model.resid, alpha = .5);
plt.plot(model.predict(df[predictors_quant]), [0 for i in range(len(df))]);
plt.title('Homoscedasticity | Quant Model, Final Model');
```



In [60]:
```
# check for the normality of the residuals
sm.qqplot(est.resid, line='s')
pylab.show()

# also check that the mean of the residuals is approx. 0.
mean_residuals = sum(est.resid)/ len(est.resid)
print("The mean of the residuals is {:.4}".format(mean_residuals))
```




```
The mean of the residuals is -7.203e-16
```

Model #10

We then took our previous model and removed certain grades as features, as they were not helping our model and possibly creating heteroscedasticity.

In [64]:
```
features = ['quant_sqft_living','quant_coffee', 'quant_schools', 'quant_scientology', 'grade_5', 'grade_6', 'grade_7', 'grade_8', 'grade_9', 'grade_10', 'grade_1
1', 'grade_12', 'grade_13', 'quant_interaction']
target = ['quant_price']
X = df[features]
y = df[target]

# running an iteration of the model using train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, random_state=1)

lm11 = LinearRegression().fit(X_train, y_train)
lm11_preds = lm11.predict(X_test)

print('R^2: ', r2_score(y_test, lm11_preds))
```

```
R^2:  0.7559686827061596
```

In [65]:
```
formula = "quant_price ~ quant_sqft_living+quant_coffee+quant_schools+quant_scientology+quant_interaction+grade_5+grade_6+grade_7+grade_8+grade_9+grade_10+grade_
11+grade_12+grade_13"
model = ols(formula= formula, data=df).fit()
model.summary()
```

Out[65]:

OLS Regression Results

Dep. Variable:	quant_price	**R-squared:**	0.761
Model:	OLS	**Adj. R-squared:**	0.761
Method:	Least Squares	**F-statistic:**	3975.
Date:	Mon, 14 Dec 2020	**Prob (F-statistic):**	0.00
Time:	16:23:30	**Log-Likelihood:**	-12316.
No. Observations:	17495	**AIC:**	2.466e+04
Df Residuals:	17480	**BIC:**	2.478e+04
Df Model:	14		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-0.7595	0.123	-6.162	0.000	-1.001	-0.518
quant_sqft_living	0.4986	0.006	89.550	0.000	0.488	0.510
quant_coffee	-0.0268	0.004	-6.779	0.000	-0.035	-0.019
quant_schools	-0.0690	0.021	-3.229	0.001	-0.111	-0.027
quant_scientology	-0.1564	0.014	-11.045	0.000	-0.184	-0.129
quant_interaction	-0.2133	0.031	-6.882	0.000	-0.274	-0.153
grade_5	0.1622	0.128	1.271	0.204	-0.088	0.412
grade_6	0.3062	0.123	2.486	0.013	0.065	0.548
grade_7	0.5827	0.123	4.730	0.000	0.341	0.824
grade_8	0.8813	0.124	7.125	0.000	0.639	1.124
grade_9	1.1946	0.125	9.592	0.000	0.951	1.439
grade_10	1.4313	0.126	11.385	0.000	1.185	1.678
grade_11	1.7186	0.129	13.371	0.000	1.467	1.971
grade_12	2.0842	0.144	14.458	0.000	1.802	2.367
grade_13	2.3268	0.236	9.839	0.000	1.863	2.790

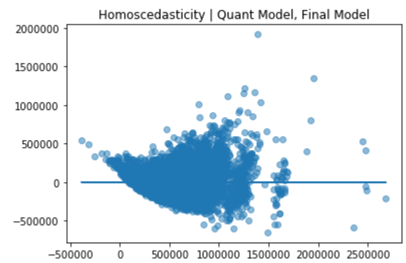
Omnibus:	391.327	**Durbin-Watson:**	1.997
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	510.627
Skew:	-0.283	**Prob(JB):**	1.31e-111
Kurtosis:	3.616	**Cond. No.**	175.

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

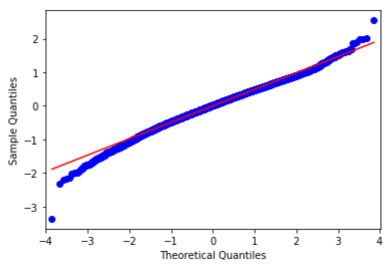
In [67]:
```
y_predict = lm10.predict(X_test)

X2 = sm.add_constant(X)

# create an OLS model
model = sm.OLS(y, X2)

# fit the data
est = model.fit()
```

```
/Users/dtunnicliffe/anaconda3/envs/learn-env/lib/python3.6/site-packages/numpy/core/fromnumeric.py:2580: FutureWarning: Method .ptp is deprecated and will be re
moved in a future version. Use numpy.ptp instead.
  return ptp(axis=axis, out=out, **kwargs)
```

In [68]:
```
f = 'price ~ quant_sqft_living+quant_coffee+quant_schools+quant_scientology+quant_interaction+grade_6+grade_7+grade_8+grade_9+grade_10+grade_11+grade_12+grade_1
3'
model = ols(formula = f, data = df).fit()

predictors_quant = ['quant_sqft_living','quant_coffee', 'quant_schools', 'quant_scientology', 'grade_5', 'grade_6', 'grade_7', 'grade_8', 'grade_9', 'grade_10',
'grade_11', 'grade_12', 'grade_13', 'quant_interaction']

plt.scatter(model.predict(df[predictors_quant]), model.resid, alpha = .5);
plt.plot(model.predict(df[predictors_quant]), [0 for i in range(len(df))]);
plt.title('Homoscedasticity | Quant Model, Final Model');
```



In [69]:
```
# check for the normality of the residuals
sm.qqplot(est.resid, line='s')
pylab.show()

# also check that the mean of the residuals is approx. 0.
mean_residuals = sum(est.resid)/ len(est.resid)
print("The mean of the residuals is {:.4}".format(mean_residuals))
```




```
The mean of the residuals is -7.203e-16
```

Our residuals are relatively normal.

Our homoscedasticity declines with this final iteration; however, our R-squared, p-values, Durbin-Watson, and prob(F-statistic) are better than they were previously.

Results

The results of our complete analysis were as follows:

- The feature with the highest impact on our R-squared value was square-footage of living space, which was positively correlated with house prices.
- The feature with the next-highest impact was distance to a top school, which was negatively correlated with house prices.
- The feature with the next-highest impact was building grade, which was positively correlated with house prices.
- The feature with the next-highest impact was distance to a scientology church, which was negatively correlated with house prices.
- The feature with the next-highest impact was distance to a great coffee shop, which was negatively correlated with house prices.
- The interaction between distance to a top school and distance to a scientology church was significant, as there was multicolinearity between the two. Accounting for this interaction showed improvement to our model.
- And finally, the feature with the least impact was distance to a park, which had no significant impact on our model.

We are confident that the results we extrapolated from this analysis would generalize beyond the data that we have. By looking at the available data, the trends and correlations we found were true for houses built from 1900 to 2015, so we are confident that they would hold true for houses built today. Despite the global pandemic, people are still buying and selling their homes. We have seen that children are still largely attending schools, and we speculate that people continue to desire a well-built homes with a large amount of living space, now more than ever. And the data has shown that people tend to pay more for a home that's near a good coffee shop and a scientology church!

If the recommendations that we made are put to use, we are confident that King County Developers will have a successful career in the housing market. From the data, it is clear that all the attributes we have discussed are correlated with high home sale prices, which is exactly what King County Developers will want for their projects.

Final Evaluation and Conclusion

Our best model had an R-squared value of 0.761, telling us that the model fit the data with an accuracy of 76%. After reviewing this final iteration, we felt confident in our recommendations that all of our available features except parks be considered by home developers in order to increase selling price. Sqare-feet of living space, building grade, distance to great schools, coffee shops, and churches of scientology, as well as the interaction between schools and scientology churches, all play a valuable role in predicting the price of a house in King County.

The prob(F-statistic) of 0.00 tells us that there is an extremely low probability of achieving these results with the null hypothesis being true, and tells us that our regression is meaningful. Our p-values for our features are well below our alpha or significance level, showing that they are each contributing to the model significantly. With an alpha of 0.05, at a confidence level of 95%, we reject the null hypothesis that there is no relationship between our features and our target variable, price.

Our recommendations are as follows:

- increase square-footage of living space
- attain the highest possible building grade
- build and develop homes in close proximity to a top school district
- build and develop homes in close proximity to a highly-rated coffee shop
- build and develop homes in close proximity to a scientology church

By following the above recommendations, a housing development company in King County can increase their chances of selling higher-priced homes.

In the future, our next steps would be reducing noise in the data to improve the accuracy of our model. Additionally, we would like to investigate certain features, such as constructional/architectural values of the house, to see what trends we could discern from that. Some ideas would be whether basements are correlated with higher house prices, or whether the amount of bathrooms has an impact.