King County Sales Assessment

In [1]:
```
# Import relevant modules
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.datasets import make_regression
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_validate, ShuffleSplit
import sklearn.metrics as metrics
from random import gauss
from mpl_toolkits.mplot3d import Axes3D
from scipy import stats as stats

%matplotlib inline
```

In [2]:
```
# Import data
df = pd.read_csv('data/kc_house_data.csv')
df.head()
```

Out[2]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront
0	7129300520	10/13/2014	221900.0	3	1.00	1180	5650	1.0	NaN
1	6414100192	12/9/2014	538000.0	3	2.25	2570	7242	2.0	NO
2	5631500400	2/25/2015	180000.0	2	1.00	770	10000	1.0	NO
3	2487200875	12/9/2014	604000.0	4	3.00	1960	5000	1.0	NO
4	1954400510	2/18/2015	510000.0	3	2.00	1680	8080	1.0	NO

5 rows × 21 columns

In [3]:
```
df.describe()
```

Out[3]:

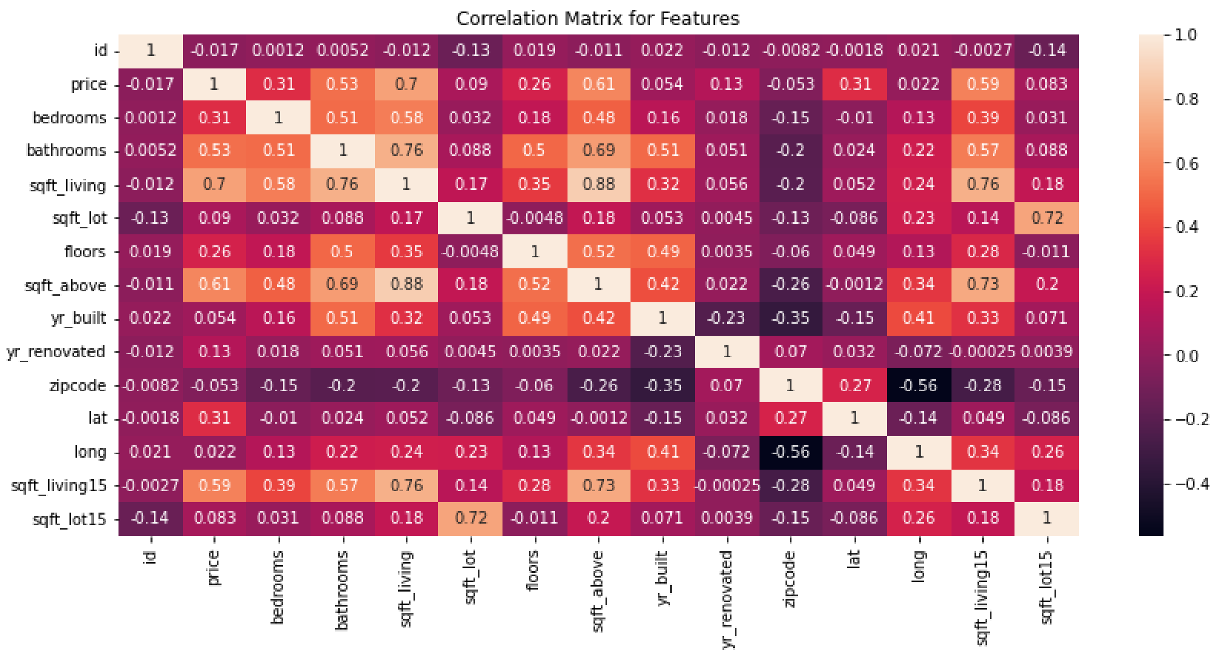
	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	
count	2.159700e+04	2.159700e+04	21597.000000	21597.000000	21597.000000	2.159700e+04	21597
mean	4.580474e+09	5.402966e+05	3.373200	2.115826	2080.321850	1.509941e+04	1
std	2.876736e+09	3.673681e+05	0.926299	0.768984	918.106125	4.141264e+04	0
min	1.000102e+06	7.800000e+04	1.000000	0.500000	370.000000	5.200000e+02	1
25%	2.123049e+09	3.220000e+05	3.000000	1.750000	1430.000000	5.040000e+03	1
50%	3.904930e+09	4.500000e+05	3.000000	2.250000	1910.000000	7.618000e+03	1
75%	7.308900e+09	6.450000e+05	4.000000	2.500000	2550.000000	1.068500e+04	2
max	9.900000e+09	7.700000e+06	33.000000	8.000000	13540.000000	1.651359e+06	3

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21597 entries, 0 to 21596
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   id             21597 non-null  int64  
 1   date           21597 non-null  object 
 2   price          21597 non-null  float64
 3   bedrooms       21597 non-null  int64  
 4   bathrooms      21597 non-null  float64
 5   sqft_living    21597 non-null  int64  
 6   sqft_lot       21597 non-null  int64  
 7   floors         21597 non-null  float64
 8   waterfront     19221 non-null  object 
 9   view           21534 non-null  object 
 10  condition      21597 non-null  object 
 11  grade          21597 non-null  object 
 12  sqft_above     21597 non-null  int64  
 13  sqft_basement  21597 non-null  object 
 14  yr_built       21597 non-null  int64  
 15  yr_renovated   17755 non-null  float64
 16  zipcode        21597 non-null  int64  
 17  lat            21597 non-null  float64
 18  long           21597 non-null  float64
 19  sqft_living15  21597 non-null  int64  
 20  sqft_lot15     21597 non-null  int64  
dtypes: float64(6), int64(9), object(6)
memory usage: 3.5+ MB
```

Baseline Models

In [5]:
```
# Lets throw up a heat map to see our simple correlation matrix
plt.figure(figsize=(14, 6))
sns.heatmap(df.corr(), annot=True)
plt.title('Correlation Matrix for Features')
plt.show()
```



In [6]:
```
# putting price in the logarithmic scale raises r squared coefficients later on,
df['l_price'] = np.log(df['price'])
```

In [7]:
```
# log scaling the living space square footage does not change the r squared much,
df['sqft_living_trans'] = np.log(df['sqft_living'])
```

In [8]:
```
# Grade is ordinal, so here we convert the column values to a 1-10 scale, rather
df['grade_final'] = df.grade.map(lambda x: int(x[0]))
# we dont use Grade in the final model, but I feel it is pertinent to show an exc
```

In [9]:
```
# Drop NA values to least common denominator to allow for most comparisons across
df_clean = df.dropna(axis = 0, how = 'any')
```

In [10]:
```
df_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15762 entries, 1 to 21596
Data columns (total 24 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   id                 15762 non-null  int64  
 1   date               15762 non-null  object 
 2   price              15762 non-null  float64
 3   bedrooms           15762 non-null  int64  
 4   bathrooms          15762 non-null  float64
 5   sqft_living        15762 non-null  int64  
 6   sqft_lot           15762 non-null  int64  
 7   floors             15762 non-null  float64
 8   waterfront         15762 non-null  object 
 9   view               15762 non-null  object 
 10  condition          15762 non-null  object 
 11  grade              15762 non-null  object 
 12  sqft_above         15762 non-null  int64  
 13  sqft_basement      15762 non-null  object 
 14  yr_built           15762 non-null  int64  
 15  yr_renovated       15762 non-null  float64
 16  zipcode            15762 non-null  int64  
 17  lat                15762 non-null  float64
 18  long               15762 non-null  float64
 19  sqft_living15      15762 non-null  int64  
 20  sqft_lot15         15762 non-null  int64  
 21  l_price            15762 non-null  float64
 22  sqft_living_trans  15762 non-null  float64
 23  grade_final        15762 non-null  int64  
dtypes: float64(8), int64(10), object(6)
memory usage: 3.0+ MB
```

In [11]:
```
# Create dummies for categorical variables
df_dum = pd.get_dummies(df_clean, columns = ['waterfront', 'zipcode'], drop_first
```

In [12]:
```
df_dum.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15762 entries, 1 to 21596
Data columns (total 92 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   id                 15762 non-null  int64  
 1   date               15762 non-null  object 
 2   price              15762 non-null  float64
 3   bedrooms           15762 non-null  int64  
 4   bathrooms          15762 non-null  float64
 5   sqft_living        15762 non-null  int64  
 6   sqft_lot           15762 non-null  int64  
 7   floors             15762 non-null  float64
 8   view               15762 non-null  object 
 9   condition          15762 non-null  object 
 10  grade              15762 non-null  object 
 11  sqft_above         15762 non-null  int64  
 12  sqft_basement      15762 non-null  object 
 13  yr_built           15762 non-null  int64  
 14  yr_renovated       15762 non-null  float64
 15  lat                15762 non-null  float64
 16  long               15762 non-null  float64
 17  sqft_living15      15762 non-null  int64  
 18  sqft_lot15         15762 non-null  int64  
 19  l_price            15762 non-null  float64
 20  sqft_living_trans  15762 non-null  float64
 21  grade_final        15762 non-null  int64  
 22  waterfront_YES     15762 non-null  uint8  
 23  zipcode_98002      15762 non-null  uint8  
 24  zipcode_98003      15762 non-null  uint8  
 25  zipcode_98004      15762 non-null  uint8  
 26  zipcode_98005      15762 non-null  uint8  
 27  zipcode_98006      15762 non-null  uint8  
 28  zipcode_98007      15762 non-null  uint8  
 29  zipcode_98008      15762 non-null  uint8  
 30  zipcode_98010      15762 non-null  uint8  
 31  zipcode_98011      15762 non-null  uint8  
 32  zipcode_98014      15762 non-null  uint8  
 33  zipcode_98019      15762 non-null  uint8  
 34  zipcode_98022      15762 non-null  uint8  
 35  zipcode_98023      15762 non-null  uint8  
 36  zipcode_98024      15762 non-null  uint8  
 37  zipcode_98027      15762 non-null  uint8  
 38  zipcode_98028      15762 non-null  uint8  
 39  zipcode_98029      15762 non-null  uint8  
 40  zipcode_98030      15762 non-null  uint8  
 41  zipcode_98031      15762 non-null  uint8  
 42  zipcode_98032      15762 non-null  uint8  
 43  zipcode_98033      15762 non-null  uint8  
 44  zipcode_98034      15762 non-null  uint8  
 45  zipcode_98038      15762 non-null  uint8  
 46  zipcode_98039      15762 non-null  uint8  
 47  zipcode_98040      15762 non-null  uint8  
 48  zipcode_98042      15762 non-null  uint8  
 49  zipcode_98045      15762 non-null  uint8  
```

```
 50  zipcode_98052      15762 non-null  uint8
 51  zipcode_98053      15762 non-null  uint8
 52  zipcode_98055      15762 non-null  uint8
 53  zipcode_98056      15762 non-null  uint8
 54  zipcode_98058      15762 non-null  uint8
 55  zipcode_98059      15762 non-null  uint8
 56  zipcode_98065      15762 non-null  uint8
 57  zipcode_98070      15762 non-null  uint8
 58  zipcode_98072      15762 non-null  uint8
 59  zipcode_98074      15762 non-null  uint8
 60  zipcode_98075      15762 non-null  uint8
 61  zipcode_98077      15762 non-null  uint8
 62  zipcode_98092      15762 non-null  uint8
 63  zipcode_98102      15762 non-null  uint8
 64  zipcode_98103      15762 non-null  uint8
 65  zipcode_98105      15762 non-null  uint8
 66  zipcode_98106      15762 non-null  uint8
 67  zipcode_98107      15762 non-null  uint8
 68  zipcode_98108      15762 non-null  uint8
 69  zipcode_98109      15762 non-null  uint8
 70  zipcode_98112      15762 non-null  uint8
 71  zipcode_98115      15762 non-null  uint8
 72  zipcode_98116      15762 non-null  uint8
 73  zipcode_98117      15762 non-null  uint8
 74  zipcode_98118      15762 non-null  uint8
 75  zipcode_98119      15762 non-null  uint8
 76  zipcode_98122      15762 non-null  uint8
 77  zipcode_98125      15762 non-null  uint8
 78  zipcode_98126      15762 non-null  uint8
 79  zipcode_98133      15762 non-null  uint8
 80  zipcode_98136      15762 non-null  uint8
 81  zipcode_98144      15762 non-null  uint8
 82  zipcode_98146      15762 non-null  uint8
 83  zipcode_98148      15762 non-null  uint8
 84  zipcode_98155      15762 non-null  uint8
 85  zipcode_98166      15762 non-null  uint8
 86  zipcode_98168      15762 non-null  uint8
 87  zipcode_98177      15762 non-null  uint8
 88  zipcode_98178      15762 non-null  uint8
 89  zipcode_98188      15762 non-null  uint8
 90  zipcode_98198      15762 non-null  uint8
 91  zipcode_98199      15762 non-null  uint8
dtypes: float64(8), int64(9), object(5), uint8(70)
memory usage: 3.8+ MB
```

In [13]:
```
df_dum.columns
```

Out[13]:
```
Index(['id', 'date', 'price', 'bedrooms', 'bathrooms', 'sqft_living',
       'sqft_lot', 'floors', 'view', 'condition', 'grade', 'sqft_above',
       'sqft_basement', 'yr_built', 'yr_renovated', 'lat', 'long',
       'sqft_living15', 'sqft_lot15', 'l_price', 'sqft_living_trans',
       'grade_final', 'waterfront_YES', 'zipcode_98002', 'zipcode_98003',
       'zipcode_98004', 'zipcode_98005', 'zipcode_98006', 'zipcode_98007',
       'zipcode_98008', 'zipcode_98010', 'zipcode_98011', 'zipcode_98014',
       'zipcode_98019', 'zipcode_98022', 'zipcode_98023', 'zipcode_98024',
       'zipcode_98027', 'zipcode_98028', 'zipcode_98029', 'zipcode_98030',
       'zipcode_98031', 'zipcode_98032', 'zipcode_98033', 'zipcode_98034',
       'zipcode_98038', 'zipcode_98039', 'zipcode_98040', 'zipcode_98042',
       'zipcode_98045', 'zipcode_98052', 'zipcode_98053', 'zipcode_98055',
       'zipcode_98056', 'zipcode_98058', 'zipcode_98059', 'zipcode_98065',
       'zipcode_98070', 'zipcode_98072', 'zipcode_98074', 'zipcode_98075',
       'zipcode_98077', 'zipcode_98092', 'zipcode_98102', 'zipcode_98103',
       'zipcode_98105', 'zipcode_98106', 'zipcode_98107', 'zipcode_98108',
       'zipcode_98109', 'zipcode_98112', 'zipcode_98115', 'zipcode_98116',
       'zipcode_98117', 'zipcode_98118', 'zipcode_98119', 'zipcode_98122',
       'zipcode_98125', 'zipcode_98126', 'zipcode_98133', 'zipcode_98136',
       'zipcode_98144', 'zipcode_98146', 'zipcode_98148', 'zipcode_98155',
       'zipcode_98166', 'zipcode_98168', 'zipcode_98177', 'zipcode_98178',
       'zipcode_98188', 'zipcode_98198', 'zipcode_98199'],
      dtype='object')
```

In [14]:
```
# Establish baseline model for strong predictor, ZIP code
X1, y1 = df_dum.drop(['id', 'date', 'price', 'bedrooms', 'bathrooms', 'sqft_livi
       'sqft_lot', 'floors', 'view', 'condition', 'grade', 'sqft_above',
       'sqft_basement', 'yr_built', 'yr_renovated', 'lat', 'long',
       'sqft_living15', 'sqft_lot15', 'l_price', 'grade_final', 'sqft_living_tra
       'waterfront_YES'], axis=1), df_dum['l_price']
X1 = sm.add_constant(X1)
baseline = sm.OLS(y1, X1).fit()
baseline.summary()
```

Out[14]:

OLS Regression Results

Dep. Variable:	l_price	**R-squared:**	0.533
Model:	OLS	**Adj. R-squared:**	0.531
Method:	Least Squares	**F-statistic:**	259.3
Date:	Fri, 24 Jun 2022	**Prob (F-statistic):**	0.00
Time:	16:55:52	**Log-Likelihood:**	-6249.8
No. Observations:	15762	**AIC:**	1.264e+04
Df Residuals:	15692	**BIC:**	1.318e+04
Df Model:	69		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	12.4593	0.023	551.867	0.000	12.415	12.504
zipcode_98002	-0.1222	0.037	-3.286	0.001	-0.195	-0.049
zipcode_98003	0.0649	0.033	1.950	0.051	-0.000	0.130
zipcode_98004	1.5554	0.033	47.497	0.000	1.491	1.620
zipcode_98005	1.1061	0.039	28.397	0.000	1.030	1.182
zipcode_98006	1.1063	0.029	37.513	0.000	1.048	1.164
zipcode_98007	0.8330	0.042	19.652	0.000	0.750	0.916
zipcode_98008	0.8220	0.033	24.593	0.000	0.756	0.887
zipcode_98010	0.4028	0.049	8.280	0.000	0.307	0.498
zipcode_98011	0.6168	0.038	16.227	0.000	0.542	0.691
zipcode_98014	0.5051	0.044	11.567	0.000	0.420	0.591
zipcode_98019	0.4711	0.039	12.063	0.000	0.395	0.548
zipcode_98022	0.1400	0.036	3.929	0.000	0.070	0.210
zipcode_98023	0.0598	0.029	2.037	0.042	0.002	0.117
zipcode_98024	0.6744	0.052	12.859	0.000	0.572	0.777
zipcode_98027	0.7855	0.031	25.678	0.000	0.726	0.845
zipcode_98028	0.5432	0.034	16.191	0.000	0.477	0.609
zipcode_98029	0.8183	0.032	25.476	0.000	0.755	0.881
zipcode_98030	0.1299	0.035	3.737	0.000	0.062	0.198

zipcode_98031	0.1292	0.034	3.809	0.000	0.063	0.196
zipcode_98032	-0.0612	0.043	-1.433	0.152	-0.145	0.023
zipcode_98033	1.0219	0.030	33.697	0.000	0.962	1.081
zipcode_98034	0.6184	0.029	21.364	0.000	0.562	0.675
zipcode_98038	0.2934	0.028	10.326	0.000	0.238	0.349
zipcode_98039	2.0415	0.064	31.805	0.000	1.916	2.167
zipcode_98040	1.4342	0.034	42.407	0.000	1.368	1.500
zipcode_98042	0.1538	0.029	5.343	0.000	0.097	0.210
zipcode_98045	0.4572	0.036	12.624	0.000	0.386	0.528
zipcode_98052	0.8831	0.029	30.801	0.000	0.827	0.939
zipcode_98053	0.8830	0.031	28.643	0.000	0.823	0.943
zipcode_98055	0.1197	0.034	3.511	0.000	0.053	0.187
zipcode_98056	0.4033	0.031	13.164	0.000	0.343	0.463
zipcode_98058	0.2606	0.030	8.640	0.000	0.201	0.320
zipcode_98059	0.5620	0.030	18.673	0.000	0.503	0.621
zipcode_98065	0.6681	0.033	20.212	0.000	0.603	0.733
zipcode_98070	0.5498	0.044	12.388	0.000	0.463	0.637
zipcode_98072	0.7376	0.034	21.752	0.000	0.671	0.804
zipcode_98074	0.9405	0.031	30.765	0.000	0.881	1.000
zipcode_98075	1.0708	0.031	34.043	0.000	1.009	1.132
zipcode_98077	0.8938	0.037	24.342	0.000	0.822	0.966
zipcode_98092	0.2122	0.032	6.680	0.000	0.150	0.275
zipcode_98102	1.1390	0.050	22.737	0.000	1.041	1.237
zipcode_98103	0.7499	0.029	26.260	0.000	0.694	0.806
zipcode_98105	1.0995	0.036	30.963	0.000	1.030	1.169
zipcode_98106	0.1653	0.033	5.066	0.000	0.101	0.229
zipcode_98107	0.7734	0.034	23.020	0.000	0.708	0.839
zipcode_98108	0.2875	0.038	7.474	0.000	0.212	0.363
zipcode_98109	1.0809	0.047	22.941	0.000	0.989	1.173
zipcode_98112	1.3419	0.034	39.837	0.000	1.276	1.408
zipcode_98115	0.8248	0.029	28.644	0.000	0.768	0.881
zipcode_98116	0.7987	0.032	24.891	0.000	0.736	0.862
zipcode_98117	0.7549	0.029	26.169	0.000	0.698	0.811
zipcode_98118	0.4030	0.029	13.719	0.000	0.345	0.461
zipcode_98119	1.1250	0.038	29.247	0.000	1.050	1.200
zipcode_98122	0.8088	0.034	24.042	0.000	0.743	0.875
zipcode_98125	0.5227	0.031	16.984	0.000	0.462	0.583

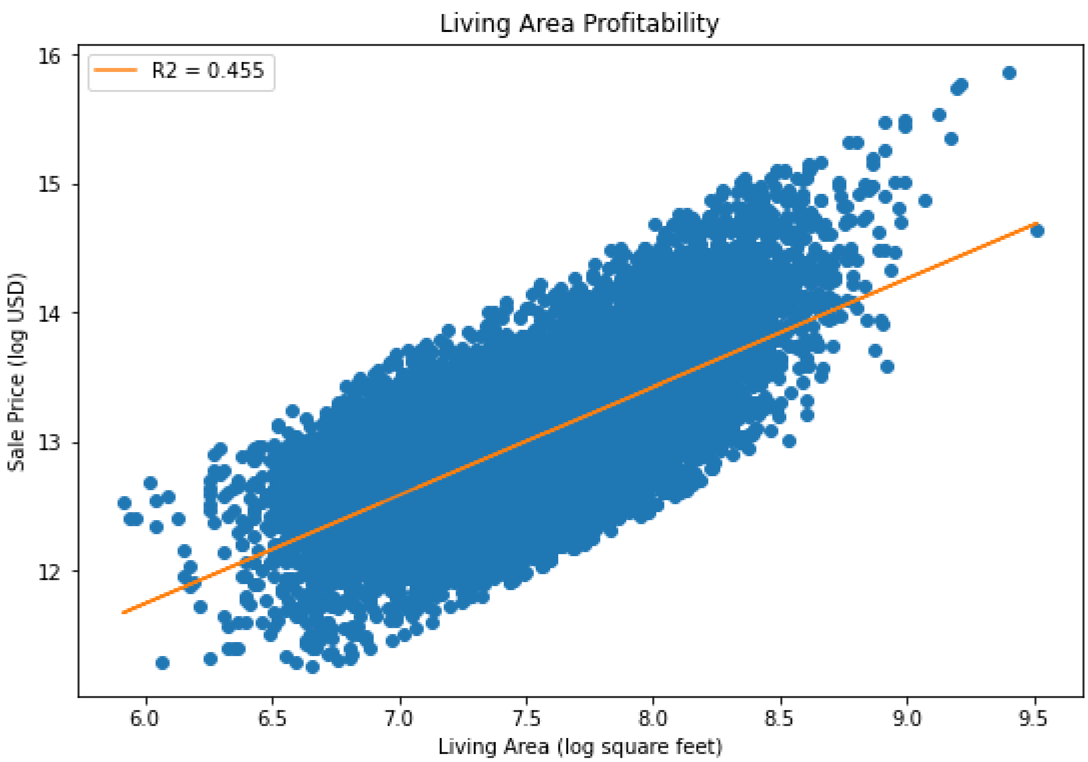
zipcode_98126	0.4448	0.032	13.890	0.000	0.382	0.508
zipcode_98133	0.3769	0.030	12.668	0.000	0.319	0.435
zipcode_98136	0.6987	0.034	20.373	0.000	0.631	0.766
zipcode_98144	0.6993	0.032	22.029	0.000	0.637	0.762
zipcode_98146	0.1927	0.033	5.767	0.000	0.127	0.258
zipcode_98148	0.0519	0.060	0.865	0.387	-0.066	0.170
zipcode_98155	0.3953	0.030	12.990	0.000	0.336	0.455
zipcode_98166	0.4791	0.035	13.759	0.000	0.411	0.547
zipcode_98168	-0.0873	0.034	-2.538	0.011	-0.155	-0.020
zipcode_98177	0.8561	0.035	24.665	0.000	0.788	0.924
zipcode_98178	0.1134	0.035	3.278	0.001	0.046	0.181
zipcode_98188	0.0582	0.043	1.353	0.176	-0.026	0.142
zipcode_98198	0.0662	0.034	1.955	0.051	-0.000	0.133
zipcode_98199	1.0240	0.033	30.904	0.000	0.959	1.089

Omnibus:	1583.149	**Durbin-Watson:**	1.985
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	3910.041
Skew:	0.594	**Prob(JB):**	0.00
Kurtosis:	5.132	**Cond. No.**	67.1

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

In [15]:
```
X2, y2 = df['sqft_living_trans'], df['l_price']
plt.figure(figsize=(9,6))

plt.plot(X2, y2, 'o')

# get m (slope) and b(intercept) of linear regression line
m, b = np.polyfit(X2, y2, 1)

# add linear regression line to scatterplot
plt.plot(X2, m*X2+b, label = 'R2 = 0.455')

plt.title('Living Area Profitability')
plt.xlabel('Living Area (log square feet)')
plt.ylabel('Sale Price (log USD)')
plt.legend()
;
```

Out[15]: ''



Final Model

In order to cut down on unnecessary code as per the guidelines, I omit the model summaries for multiple regressions that included grade and waterfront status in favor of the final model. Adding grade as an ordinal predictor does not increase the accuracy of the model at all, and mildly increases skewness. I do not currently

In [16]:
```
# Assign final predictors and dependent variable
X, y = df_dum.drop(['id', 'date', 'price', 'bedrooms', 'bathrooms',
       'sqft_lot', 'floors', 'view', 'condition', 'grade', 'sqft_above',
       'sqft_basement', 'yr_built', 'yr_renovated', 'lat', 'long',
       'sqft_living15', 'sqft_lot15', 'l_price',
            'sqft_living', 'grade_final'
       ], axis=1), df_dum['l_price']
```

In [17]:
```
X = sm.add_constant(X)
model1 = sm.OLS(y, X).fit()
model1.summary()
```

Out[17]:

OLS Regression Results

Dep. Variable:	l_price	**R-squared:**	0.833
Model:	OLS	**Adj. R-squared:**	0.832
Method:	Least Squares	**F-statistic:**	1100.
Date:	Fri, 24 Jun 2022	**Prob (F-statistic):**	0.00
Time:	16:55:53	**Log-Likelihood:**	1842.6
No. Observations:	15762	**AIC:**	-3541.
Df Residuals:	15690	**BIC:**	-2989.
Df Model:	71		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	7.0687	0.036	193.898	0.000	6.997	7.140

In [18]:
```
# Prepare train and test data from X and y variables for cross validation
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_s

linreg = LinearRegression()
linreg.fit(X_train, y_train)

y_hat_train = linreg.predict(X_train)
y_hat_test = linreg.predict(X_test)
```

In [19]:
```
# Inspect residuals
train_residuals = y_hat_train - y_train
test_residuals = y_hat_test - y_test
print(train_residuals)
print(test_residuals)
```

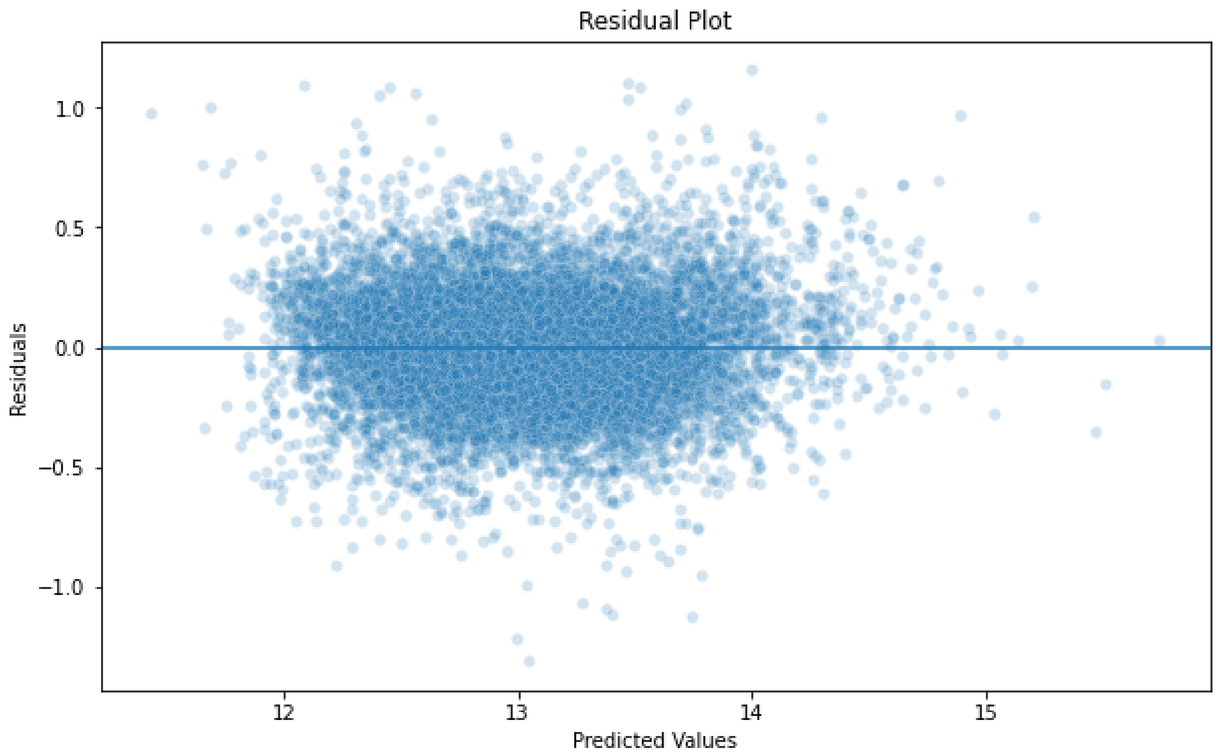
```
18060    0.091455
4178    -0.018608
13219    0.082941
2839    -0.266770
5520     0.256757
           ...
7129     0.031156
18336    0.246032
7384    -0.242698
1159    -0.580698
9955     0.115760
Name: l_price, Length: 12609, dtype: float64
8446     0.379067
7473    -0.238393
20534    0.142403
5063     0.147336
17989   -0.215431
           ...
2330    -0.024277
10260    0.106158
16786    0.203038
8159     0.079904
10136    0.255806
Name: l_price, Length: 3153, dtype: float64
```

In [20]:
```
# Determine MSE
mse_train = np.sum((y_train-y_hat_train)**2)/len(y_train)
mse_test = np.sum((y_test-y_hat_test)**2)/len(y_test)
print('Train Mean Squarred Error:', mse_train)
print('Test Mean Squarred Error:', mse_test)
```

```
Train Mean Squarred Error: 0.04624036654907834
Test Mean Squarred Error: 0.04704959588781619
```

In [21]:
```
# Create a residual plot to explore possible patterns
df_dum["predicted"] = model1.predict(X)
df_dum["residuals"] = model1.resid

plt.figure(figsize=(10, 6))
sns.scatterplot(data=df_dum, x="predicted", y="residuals", alpha = 0.2)
plt.axhline(y=0)
plt.title('Residual Plot')
plt.xlabel('Predicted Values')
plt.ylabel('Residuals')
;
```

Out[21]: ''



In [22]:
```
splitter = ShuffleSplit(n_splits=3, test_size=0.25, random_state=0)

baseline_scores = cross_validate(estimator=linreg, X=X_train, y=y_train, return_t

print("Train score:     ", baseline_scores["train_score"].mean())
print("Validation score:", baseline_scores["test_score"].mean())
```

```
Train score:      0.8354967482802792
Validation score: 0.8314829925222368
```