King County Housing with Multiple Linear Regression

Authors: Diane Tunnicliffe, Dana Rausch, Matthew Lipman

Notebook 2: Data Preparation

This notebook contains a breakdown of the step-by-step processes that we used to co
scrub, and transform our data. It includes variations of narrowing our scope and explor
the impacts that our different transformations have on the data. For the actual full proc
the data was obtained, and a full description of each data set, please see our first note
'business_problem_and_data_understanding'.

In [172]:
```
 importing the packages we will be using for this project
mport pandas as pd
 setting pandas display to avoid scientific notation in my datafr
d.options.display.float_format = '{:.2f}'.format
mport numpy as np
mport matplotlib.pyplot as plt
mport seaborn as sns
mport sklearn

rom bs4 import BeautifulSoup
mport json
mport requests

mport folium

mport haversine as hs

mport statsmodels.api as sm
rom statsmodels.formula.api import ols
rom statsmodels.stats import diagnostic as diag
rom statsmodels.stats.outliers_influence import variance_inflatic

rom sklearn.metrics import r2_score
rom sklearn.linear_model import LinearRegression
rom sklearn.neighbors import NearestNeighbors
rom sklearn.model_selection import train_test_split
rom sklearn.metrics import mean_squared_error, r2_score, mean_abs

mport scipy.stats as stats

mport pylab

matplotlib inline
```

King County Houses

In [173]:

```
# reading the csv file
df = pd.read_csv('data/kc_house_data.csv')
# previewing the DataFrame
df.head()
```

Out[173]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floo
0	7129300520	10/13/2014	221900.00	3	1.00	1180	5650	1.(
1	6414100192	12/9/2014	538000.00	3	2.25	2570	7242	2.(
2	5631500400	2/25/2015	180000.00	2	1.00	770	10000	1.(
3	2487200875	12/9/2014	604000.00	4	3.00	1960	5000	1.(
4	1954400510	2/18/2015	510000.00	3	2.00	1680	8080	1.(

5 rows × 21 columns

In [174]:

```
# generating descriptive statistics
df.describe()
```

Out[174]:

living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	yr_
97.00	21597.00	21597.00	19221.00	21534.00	21597.00	21597.00	21597.00	2159
80.32	15099.41	1.49	0.01	0.23	3.41	7.66	1788.60	197
18.11	41412.64	0.54	0.09	0.77	0.65	1.17	827.76	2
70.00	520.00	1.00	0.00	0.00	1.00	3.00	370.00	190
30.00	5040.00	1.00	0.00	0.00	3.00	7.00	1190.00	195
10.00	7618.00	1.50	0.00	0.00	3.00	7.00	1560.00	197
50.00	10685.00	2.00	0.00	0.00	4.00	8.00	2210.00	199
40.00	1651359.00	3.50	1.00	4.00	5.00	13.00	9410.00	201

In [175]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21597 entries, 0 to 21596
Data columns (total 21 columns):
id               21597 non-null int64
date             21597 non-null object
price            21597 non-null float64
bedrooms         21597 non-null int64
bathrooms        21597 non-null float64
sqft_living      21597 non-null int64
sqft_lot         21597 non-null int64
floors           21597 non-null float64
waterfront       19221 non-null float64
view             21534 non-null float64
condition        21597 non-null int64
grade            21597 non-null int64
sqft_above       21597 non-null int64
sqft_basement    21597 non-null object
yr_built         21597 non-null int64
yr_renovated     17755 non-null float64
zipcode          21597 non-null int64
lat              21597 non-null float64
long             21597 non-null float64
sqft_living15    21597 non-null int64
sqft_lot15       21597 non-null int64
dtypes: float64(8), int64(11), object(2)
memory usage: 3.5+ MB
```

Narrowing down our price range

In [176]:
```
std = df.price.std()
print('std: ',std)
mean = df.price.mean()
print('mean: ', mean)
std_1 = mean + std
std_1m = mean - std
print('mean +1 std: ',std_1)
print('mean -1 std: ',std_1m)
```

```
std:  367368.1401013945
mean:  540296.5735055795
mean +1 std:  907664.713606974
mean -1 std:  172928.433404185
```

In [177]:
```
df = df.loc[(df['price']<std_1) & (df['price']>std_1m)]
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19205 entries, 0 to 21596
Data columns (total 21 columns):
id               19205 non-null int64
date             19205 non-null object
price            19205 non-null float64
bedrooms         19205 non-null int64
bathrooms        19205 non-null float64
sqft_living      19205 non-null int64
sqft_lot         19205 non-null int64
floors           19205 non-null float64
waterfront       17082 non-null float64
view             19149 non-null float64
condition        19205 non-null int64
grade            19205 non-null int64
sqft_above       19205 non-null int64
sqft_basement    19205 non-null object
yr_built         19205 non-null int64
yr_renovated     15798 non-null float64
zipcode          19205 non-null int64
lat              19205 non-null float64
long             19205 non-null float64
sqft_living15    19205 non-null int64
sqft_lot15       19205 non-null int64
dtypes: float64(8), int64(11), object(2)
memory usage: 3.2+ MB
```

In [178]:
```
#dropping unnecessary columns
drop = ['id','date', 'yr_built', 'bedrooms', 'bathrooms','sqft_lo
df = df.drop(columns = drop, axis=1)
```

In [179]:
```
df.columns
```

Out[179]:
```
Index(['price', 'sqft_living', 'grade', 'lat', 'long'], dtype='ob
```

In [180]:
```
df.isnull().sum()
```

Out[180]:
```
price          0
sqft_living    0
grade          0
lat            0
long           0
dtype: int64
```

King County Parks

In [181]:
```
# importing park data
# reading the csv file
king_parks = pd.read_csv('data/ParkAddresses_wLatLong.csv', inde
# previewing the DataFrame
king_parks.head()
```

Out[181]:

	Address	Combined	
ID			
0.00	Auburn Black Diamond Rd and SE Green Valley Rd...	47.301182311345315, -122.17491469179195	47
1.00	NE 165th St and 179th Pl NE Redmond WA 98072	47.74702351303733, -122.09810603412113	47
2.00	NaN	NaN	n
3.00	NE 138th and Juanita Drive NE Kirkland WA 98028	47.72417796430824, -122.2384511052857	47
4.00	S 284th Pl and 37th Ave S Federal Way WA 98003	47.34814028865613, -122.2811067550002	47

In [182]:
```
#create function to find distances between all points in DF and
def find_distance(dataframe):
    """
    Calculates distance between points of interest and houses.

    Generates a distance matrix for distances between houses and
    Calculates distance from each point in dataframe (df) to poi
    Converts latitude and longitude to radians in order to calcu
    Returns values as kilometers.

    Parameters:
    dataframe (Pandas DataFrame object): user input name of Panda

    Returns:
    Matrix of distances.

    """
    dist = sklearn.neighbors.DistanceMetric.get_metric('haversine

    #convert lat and long to radians
    dataframe[['lat_radians','long_radians']] = (np.radians(data

    #create list matrix (results in km)
    dist_matrix = (dist.pairwise
    (df[['lat_radians_A','long_radians_A']],
     dataframe[['lat_radians','long_radians']])*6371)

    #return a matrix DataFrame
    return pd.DataFrame(dist_matrix)
```

In [183]:
```
#convert lat and long to radians in housing data
df[['lat_radians_A','long_radians_A']] = (np.radians(df.loc[:,['
```

In [184]:
```
park_matrix = find_distance(king_parks)
```

In [185]:
```
#find min distance in each row
park_min_matrix = park_matrix.where(park_matrix.values == park_ma
    axis=1)[:,None]).drop_duplicates()
```

In [186]:
```
#create a new column with only min distance and remove the rest
park_min_matrix['min_dist_park'] = park_min_matrix[park_min_matr
    lambda x: ','.join(x.dropna().astype(str)),
    axis=1)
nearest_park = park_min_matrix['min_dist_park']
```

In [187]:
```
df = df.join(nearest_park)
```

In [188]:
```
df.head()
```

Out[188]:

	price	sqft_living	grade	lat	long	lat_radians_A	long_radians_A	n
0	221900.00	1180	7	47.51	-122.26	0.83	-2.13	2.0383
1	538000.00	2570	7	47.72	-122.32	0.83	-2.13	5.0520
2	180000.00	770	6	47.74	-122.23	0.83	-2.13	1.3379
3	604000.00	1960	7	47.52	-122.39	0.83	-2.14	2.4485
4	510000.00	1680	8	47.62	-122.05	0.83	-2.13	2.67283

In [189]:
```
df['min_dist_park']= df['min_dist_park'].astype('float64')
```

King County Top Schools

In [190]:
```python
# importing school data
# for entire data obtaining process, please see other notebook

# reading the csv file
top_schools_df = pd.read_csv('data/top_schools.csv')
# previewing the DataFrame
top_schools_df.head()
```

Out[190]:

	Unnamed: 0	year	ncessch	school_name	state_name	lea_name	zip_location
0	43	2015	530039000058	Ardmore Elementary School	Washington	Bellevue School District	98008
1	44	2015	530039000060	Bellevue High School	Washington	Bellevue School District	98004
2	45	2015	530039000062	Bennett Elementary School	Washington	Bellevue School District	98008
3	46	2015	530039000063	Cherry Crest Elementary School	Washington	Bellevue School District	98005
4	47	2015	530039000064	Chinook Middle School	Washington	Bellevue School District	98004

In [191]:
```python
top_schools_df.drop(columns = 'Unnamed: 0', axis=1, inplace=True
```

In [192]:
```python
top_schools_df.head()
```

Out[192]:

	year	ncessch	school_name	state_name	lea_name	zip_location	latitude	lo
0	2015	530039000058	Ardmore Elementary School	Washington	Bellevue School District	98008	47.64	
1	2015	530039000060	Bellevue High School	Washington	Bellevue School District	98004	47.60	
2	2015	530039000062	Bennett Elementary School	Washington	Bellevue School District	98008	47.62	
3	2015	530039000063	Cherry Crest Elementary School	Washington	Bellevue School District	98005	47.64	
4	2015	530039000064	Chinook Middle School	Washington	Bellevue School District	98004	47.63	

In [193]:
```
#geographic distance calculator
#function that identifies the distance between a point of intere
def distance_to(point_of_interest):
    """
    Calculates distance between point of interest and a house.

    Takes in coordinates for point of interest as latitude and lo
    Calculates distance from each point in dataframe (df) to poi
    Uses haversine formula to calculate distance and return as k
    Can set distances as new column of dataframe by using df['new

    Parameters:
    point_of_interest (float): user input coordinates (latitude,l

    Returns:
    Distances in kilometers, using haversine formula.

    """
    distance = df[['lat','long']].apply(lambda x: hs.haversine(x
    return distance
```

In [194]:
```
top_school_coordinates = []
x = round(top_schools_df.latitude, 2)
y = round(top_schools_df.longitude, 2)
top_school_coordinates = list(zip(x,y))
```

In [195]:
```
for i in range(len(top_school_coordinates)):
    df[f'top_school_{i}'] = distance_to(top_school_coordinates[i

top_school_cols = []
for i in range(len(top_school_coordinates)):
    top_school_cols.append(f'top_school_{i}')
    df['closest_distance_to_top_school'] = df[top_school_cols].mi
```

In [196]:
```
df.drop(columns = top_school_cols, axis=1, inplace=True)
rad_cols = ['lat_radians_A', 'long_radians_A']
df.drop(columns=rad_cols, axis=1, inplace=True)
df.head()
```

Out[196]:

	price	sqft_living	grade	lat	long	min_dist_park	closest_distance_to_top_
0	221900.00	1180	7	47.51	-122.26	2.04	
1	538000.00	2570	7	47.72	-122.32	5.05	
2	180000.00	770	6	47.74	-122.23	1.34	
3	604000.00	1960	7	47.52	-122.39	2.45	
4	510000.00	1680	8	47.62	-122.05	2.67	

King County Top 10 Coffee Shops

In [197]:
```
def get_keys(path):
    """Retrieves API key from files as api_key."""
    with open(path) as f:
        return json.load(f)

keys = get_keys("/Users/dtunnicliffe/.secret/yelp_api.json")
api_key = keys['api_key']

term = 'coffee'
location = 'King County, WA'
SEARCH_LIMIT = 10
espresso = pd.DataFrame([])
def yelp(term, location, SEARCH_LIMIT):
    """
    Creates a new dataframe of information retrieved from yelp AI

    Searches businesses and returns top results based on criteri
    Makes API call as if searching on yelp.
    Returns relevant information for businesses such as name, lo

    Parameters:
    term (str): user input term to search for.
    location (str): user input city, state, or zip code to searcl
    SEARCH_LIMIT (int): user input number of results to return.

    Returns:
    New dataframe populated with requested information.

    """
    global espresso
    url = 'https://api.yelp.com/v3/businesses/search'
    headers = {
    'Authorization': f'Bearer {api_key}',
    }
    url_params = {
    'term': term.replace(' ', '+'),
    'location': location.replace(' ', '+'),
    'limit': SEARCH_LIMIT,
    'sort_by': 'rating'
    }
    response = requests.get(url, headers=headers, params=url_para
    df_temp = pd.DataFrame.from_dict(response.json()['businesses
    espresso = espresso.append(df_temp)
    return espresso
```

In [198]:
```
espresso = yelp(term, location, SEARCH_LIMIT)
```

In [199]:
```
espresso.head()
```

Out[199]:

	id	alias	name	im
0	S6CXIQ5KrMpTPZf1eNMa2w	five-stones-coffee-company-redmond	Five Stones Coffee Company	htt media3.fl.yelpcdn.com/bphoto/Om
1	EWqgeiGor-aVJIMLc8iSKw	boon-boona-coffee-renton	Boon Boona Coffee	htt media3.fl.yelpcdn.com/bphoto/tV
2	v7xfqk9f7N8A98AQ2kddWg	anchorhead-coffee-bellevue-3	Anchorhead Coffee	htt media3.fl.yelpcdn.com/bphoto/Er
3	t2DOOFh-oJLddtpxbVlDrQ	huxdotter-coffee-north-bend	Huxdotter Coffee	htt media3.fl.yelpcdn.com/bphoto/M
4	-MzbuOLr2kAoqlQY8w7ECA	pioneer-coffee-north-bend-north-bend	Pioneer Coffee - North Bend	htt media3.fl.yelpcdn.com/bphoto/5

In [200]:
```
great_coffee_coordinates = []
x = [round(coordinate['latitude'], 2) for coordinate in espresso
y = [round(coordinate['longitude'], 2) for coordinate in espress
great_coffee_coordinates = list(zip(x,y))
```

In [201]:
```
for i in range(len(great_coffee_coordinates)):
    df[f'great_coffee_{i}'] = distance_to(great_coffee_coordinate

great_coffee_cols = []
for i in range(len(great_coffee_coordinates)):
    great_coffee_cols.append(f'great_coffee_{i}')
    df['closest_distance_to_great_coffee'] = df[great_coffee_col
```

In [202]:
```
#dropping unnecessary columns
df = df.drop(columns = great_coffee_cols, axis=1)
df.head()
```

Out[202]:

	price	sqft_living	grade	lat	long	min_dist_park	closest_distance_to_top_
0	221900.00	1180	7	47.51	-122.26	2.04	
1	538000.00	2570	7	47.72	-122.32	5.05	
2	180000.00	770	6	47.74	-122.23	1.34	
3	604000.00	1960	7	47.52	-122.39	2.45	
4	510000.00	1680	8	47.62	-122.05	2.67	

King County Churches of Scientology

In [203]:
```
#locations pulled from scientology-seattle.org
church_of_scientology_mission = (47.818100, -122.315430)
church_of_scientology_washington = (47.622380, -122.361020)
church_of_scientology_life_improvement_center = (47.615060, -122
```

In [204]:
```
#function that identifies the distance between a point of interes
def distance_to(point_of_interest):
    """
    Calculates distance between point of interest and a house.

    Takes in coordinates for point of interest as latitude and lo
    Calculates distance from each point in dataframe (df) to poin
    Uses haversine formula to calculate distance and return as ki
    Can set distances as new column of dataframe by using df['new

    Parameters:
    point_of_interest (float): user input coordinates (latitude,l

    Returns:
    Distances in kilometers, using haversine formula.

    """
    distance = df[['lat','long']].apply(lambda x: hs.haversine(x.
    return distance
```

In [205]:
```
#creating new columns of distances from houses to point of intere
df['distance_to_scientology_m'] = distance_to(church_of_scientolo
df['distance_to_scientology_w'] = distance_to(church_of_scientolo
df['distance_to_scientology_l'] = distance_to(church_of_scientolo
df['closest_distance_to_scientology'] = df[['distance_to_sciento
                                            'distance
                                            'distance
```

In [206]:
```
sci_cols = ['distance_to_scientology_m', 'distance_to_scientology
            'distance_to_scientology_l']
df.drop(columns = sci_cols, axis=1, inplace=True)
```

In [207]:
```
df.head()
```

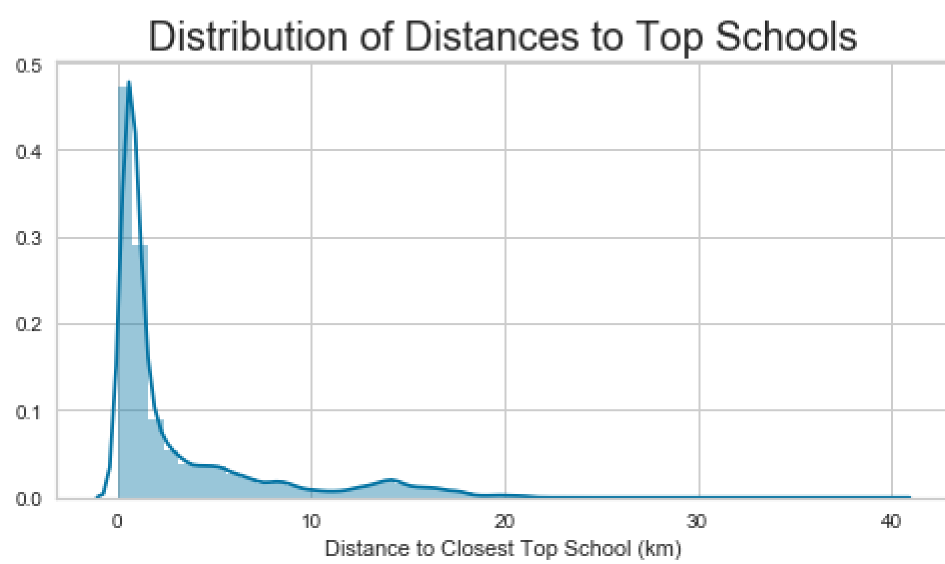
Out[207]:

park	closest_distance_to_top_school	closest_distance_to_great_coffee	closest_distance_t
2.04	0.26	4.95	
5.05	0.68	14.81	
1.34	2.00	10.63	
2.45	1.73	14.48	
2.67	1.18	8.55	

Log-Transforming Features

In [208]:
```
plt.figure(figsize=(8,4))
sns.distplot(df['closest_distance_to_top_school'])
plt.title("Distribution of Distances to Top Schools", fontsize=2
plt.xlabel('Distance to Closest Top School (km)');
print("Skewness:", df['closest_distance_to_top_school'].skew())
print("Kurtosis:", df['closest_distance_to_top_school'].kurt())
```

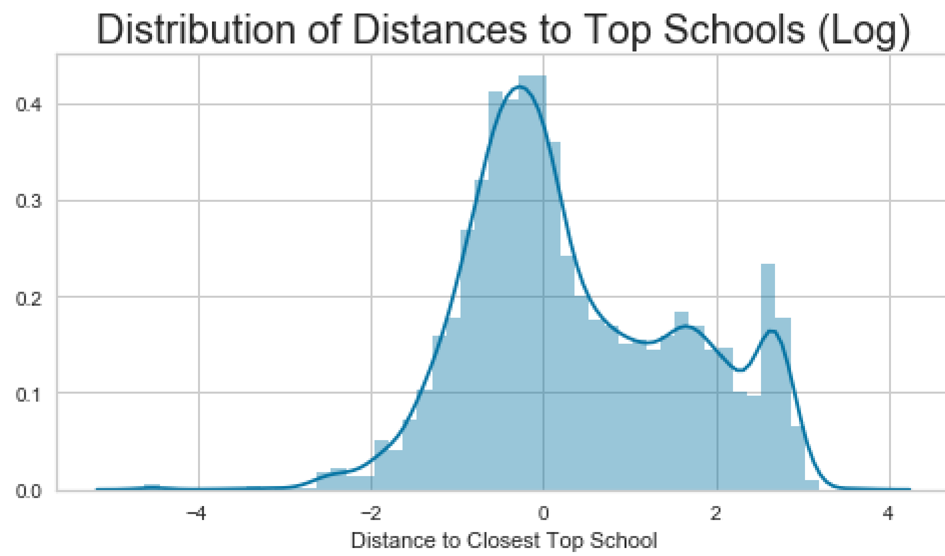
```
Skewness: 2.07081534646944
Kurtosis: 4.115792045291801
```



In [209]:
```
# removing zeroes for log transformation
df.loc[df['closest_distance_to_top_school']==0.00, 'closest_dista
#natural log transformation for 'closest_distance_to_top_school'
df['log_school'] = df['closest_distance_to_top_school'].map(lamb
```

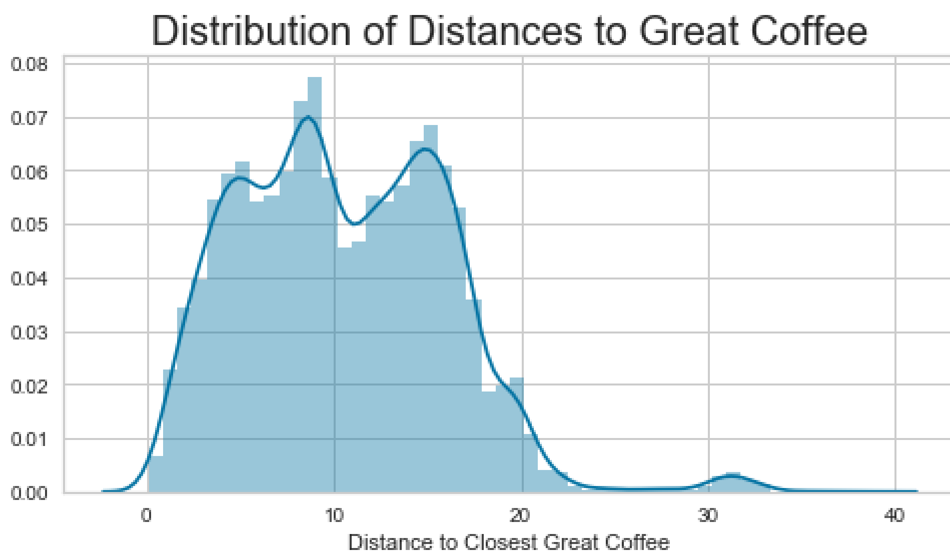
In [210]:
```
plt.figure(figsize=(8,4))
sns.distplot(df['log_school'])
plt.title("Distribution of Distances to Top Schools (Log)", fonts
plt.xlabel('Distance to Closest Top School');
print("Skewness:", df['log_school'].skew())
print("Kurtosis:", df['log_school'].kurt())
```

```
Skewness: 0.31498656015781384
Kurtosis: -0.4837932278849535
```



In [211]:
```
plt.figure(figsize=(8,4))
sns.distplot(df['closest_distance_to_great_coffee'])
plt.title("Distribution of Distances to Great Coffee", fontsize=
plt.xlabel('Distance to Closest Great Coffee');
print("Skewness:", df['closest_distance_to_great_coffee'].skew()
print("Kurtosis:", df['closest_distance_to_great_coffee'].kurt()
```

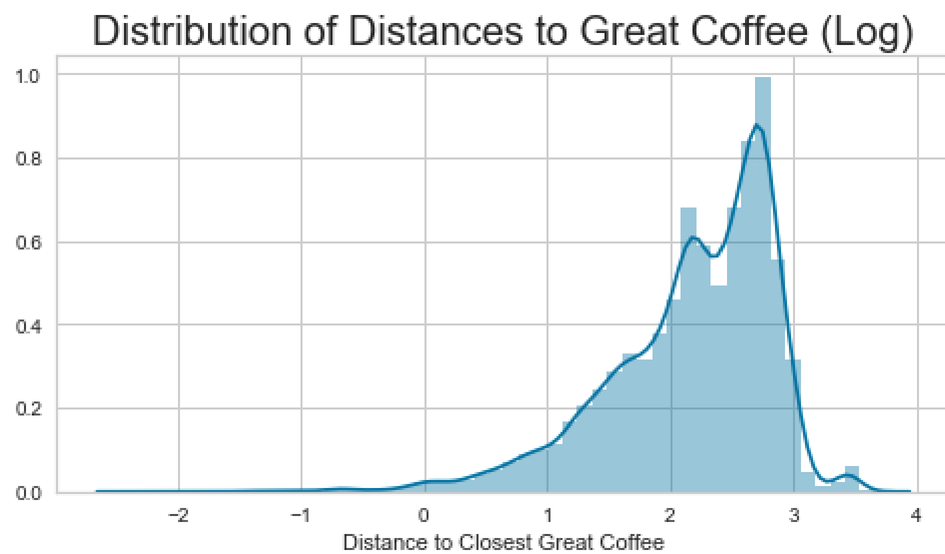
```
Skewness: 0.5617274905372351
Kurtosis: 0.7125009157364359
```



In [212]:
```
# removing zeroes for log transformation
df.loc[df['closest_distance_to_great_coffee']==0.00, 'closest_di
#natural log transformation for 'closest_distance_to_great_coffe
df['log_coffee'] = df['closest_distance_to_great_coffee'].map(la
```

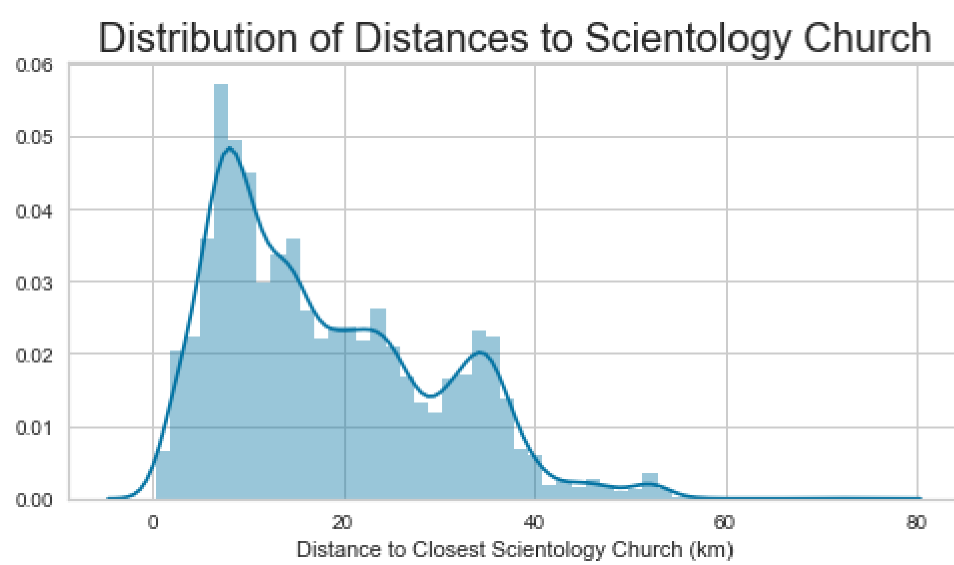
In [213]:
```
plt.figure(figsize=(8,4))
sns.distplot(df['log_coffee'])
plt.title("Distribution of Distances to Great Coffee (Log)", font
plt.xlabel('Distance to Closest Great Coffee');
print("Skewness:", df['log_coffee'].skew())
print("Kurtosis:", df['log_coffee'].kurt())
```

```
Skewness: -1.1714703303238783
Kurtosis: 1.9350322960834787
```



In [214]:
```
plt.figure(figsize=(8,4))
sns.distplot(df['closest_distance_to_scientology'])
plt.title("Distribution of Distances to Scientology Church", font
plt.xlabel('Distance to Closest Scientology Church (km)');
print("Skewness:", df['closest_distance_to_scientology'].skew())
print("Kurtosis:", df['closest_distance_to_scientology'].kurt())
```

```
Skewness: 0.729624297126709
Kurtosis: -0.13070775209001573
```



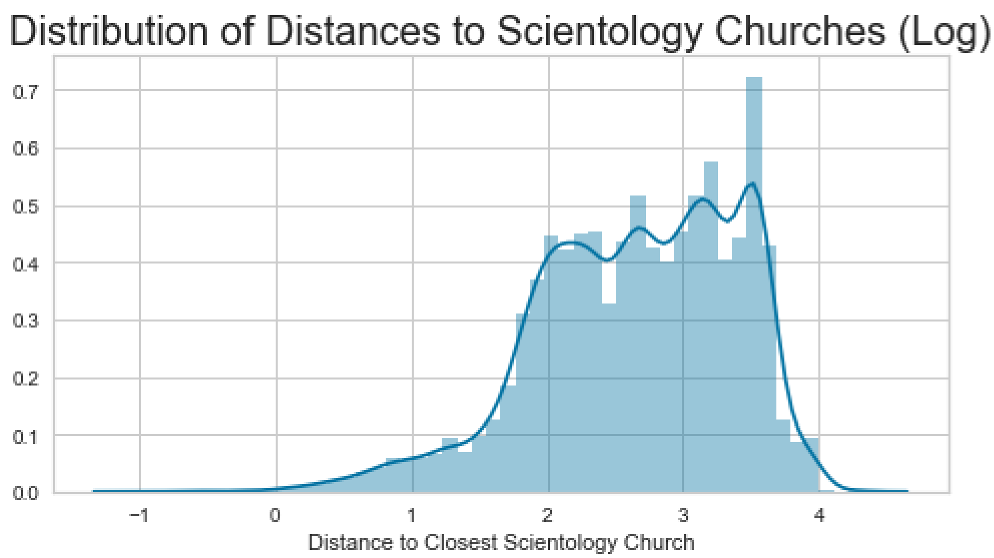
In [215]:

```
# removing zeroes for log transformation
df.loc[df['closest_distance_to_scientology']==0.00, 'closest_dist
#natural log transformation for 'closest_distance_to_scientology
df['log_scientology'] = df['closest_distance_to_scientology'].ma
```

In [216]:

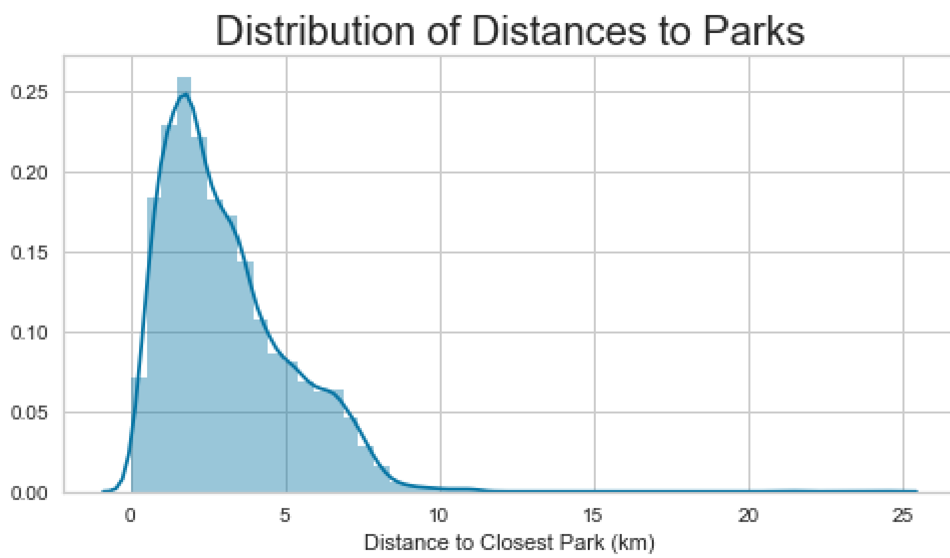
```
plt.figure(figsize=(8,4))
sns.distplot(df['log_scientology'])
plt.title("Distribution of Distances to Scientology Churches (Lo
plt.xlabel('Distance to Closest Scientology Church');
print("Skewness:", df['log_scientology'].skew())
print("Kurtosis:", df['log_scientology'].kurt())
```

```
Skewness: -0.6186336629179573
Kurtosis: 0.16752897590293658
```



In [217]:
```
plt.figure(figsize=(8,4))
sns.distplot(df['min_dist_park'])
plt.title("Distribution of Distances to Parks", fontsize=20)
plt.xlabel('Distance to Closest Park (km)');
print("Skewness:", df['min_dist_park'].skew())
print("Kurtosis:", df['min_dist_park'].kurt())
```

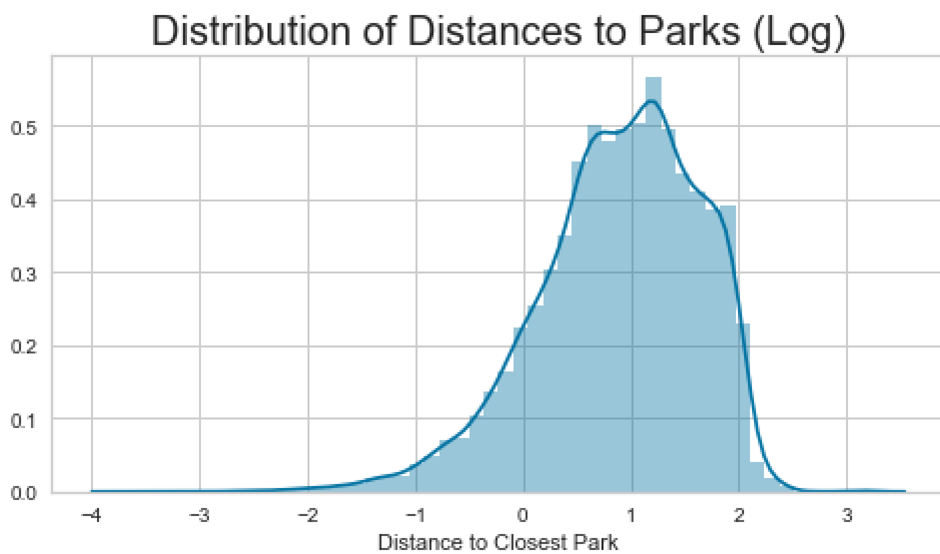
```
Skewness: 1.205427367383708
Kurtosis: 3.9928978255283716
```



In [218]:
```
# removing zeroes for log transformation
df.loc[df['min_dist_park']==0.00, 'min_dist_park']=0.01
#natural log transformation for 'min_dist_park'.
df['log_park'] = df['min_dist_park'].map(lambda x: np.log(x))
```

In [219]:
```
plt.figure(figsize=(8,4))
sns.distplot(df['log_park'])
plt.title("Distribution of Distances to Parks (Log)", fontsize=2(
plt.xlabel('Distance to Closest Park');
print("Skewness:", df['log_park'].skew())
print("Kurtosis:", df['log_park'].kurt())
```

```
Skewness: -0.697074959578087
Kurtosis: 0.6535881306866189
```



In [220]:
```
df.isnull().sum()
```

Out[220]:
```
price                                0
sqft_living                          0
grade                                0
lat                                  0
long                                 0
min_dist_park                     2712
closest_distance_to_top_school       0
closest_distance_to_great_coffee     0
closest_distance_to_scientology      0
log_school                           0
log_coffee                           0
log_scientology                      0
log_park                          2712
dtype: int64
```

In [221]:
```
df.dropna(inplace=True)
df.isnull().sum()
```

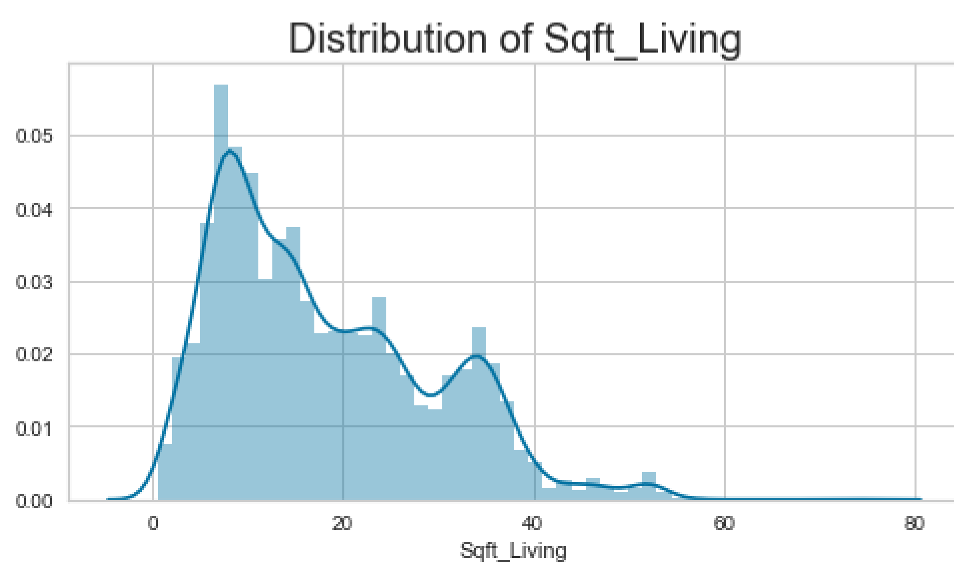
Out[221]:
```
price                                0
sqft_living                          0
grade                                0
lat                                  0
long                                 0
min_dist_park                        0
closest_distance_to_top_school       0
closest_distance_to_great_coffee     0
closest_distance_to_scientology      0
log_school                           0
log_coffee                           0
log_scientology                      0
log_park                             0
dtype: int64
```

In [222]:
```
# saving copy of DataFrame as csv file
#df.to_csv('./data/all_features_with_logs.csv')
```

Quantile Tranformation

In [241]:
```
plt.figure(figsize=(8,4))
sns.distplot(df['Closest_Distance_to_Scientology'])
plt.title("Closest Distance to Scientology", fontsize=20)
plt.xlabel('Closest Distance to Scientology');
print("Skewness:", df['closest_distance_to_scientology'].skew())
print("Kurtosis:", df['closest_distance_to_scientology'].kurt())
```

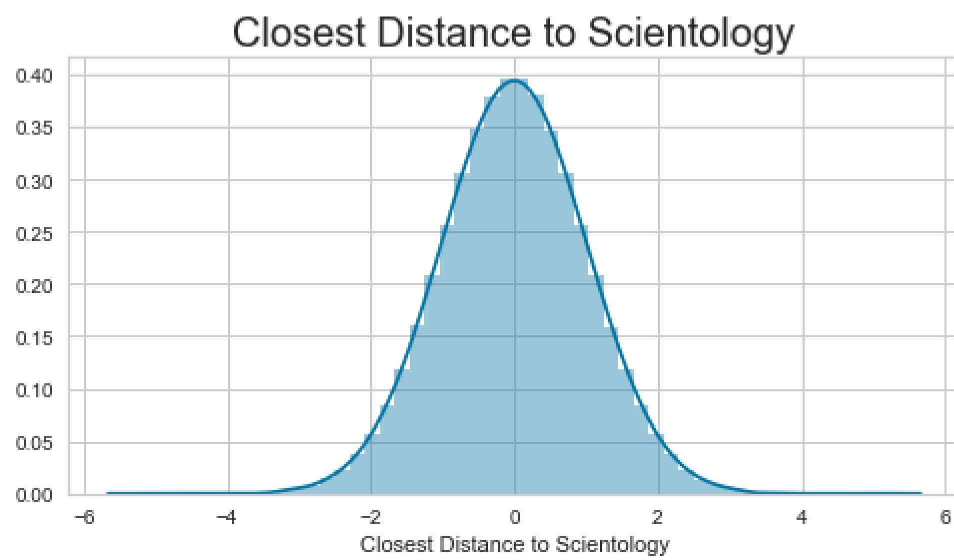
```
Skewness: 0.7651743646573521
Kurtosis: 0.8677398638009155
```



In [242]:
```
from sklearn.preprocessing import QuantileTransformer
qt = QuantileTransformer(output_distribution='normal')
to_transform= ['sqft_living', 'closest_distance_to_great_coffee'
df[to_transform] = qt.fit_transform(df[to_transform])
```

In [255]:
```
plt.figure(figsize=(8,4))
sns.distplot(df['closest_distance_to_scientology'])
plt.title("Closest Distance to Scientology (Quant)", fontsize=20
plt.xlabel('Closest Distance to Scientology');
print("Skewness:", df['closest_distance_to_scientology'].skew())
print("Kurtosis:", df['closest_distance_to_scientology'].kurt())
```

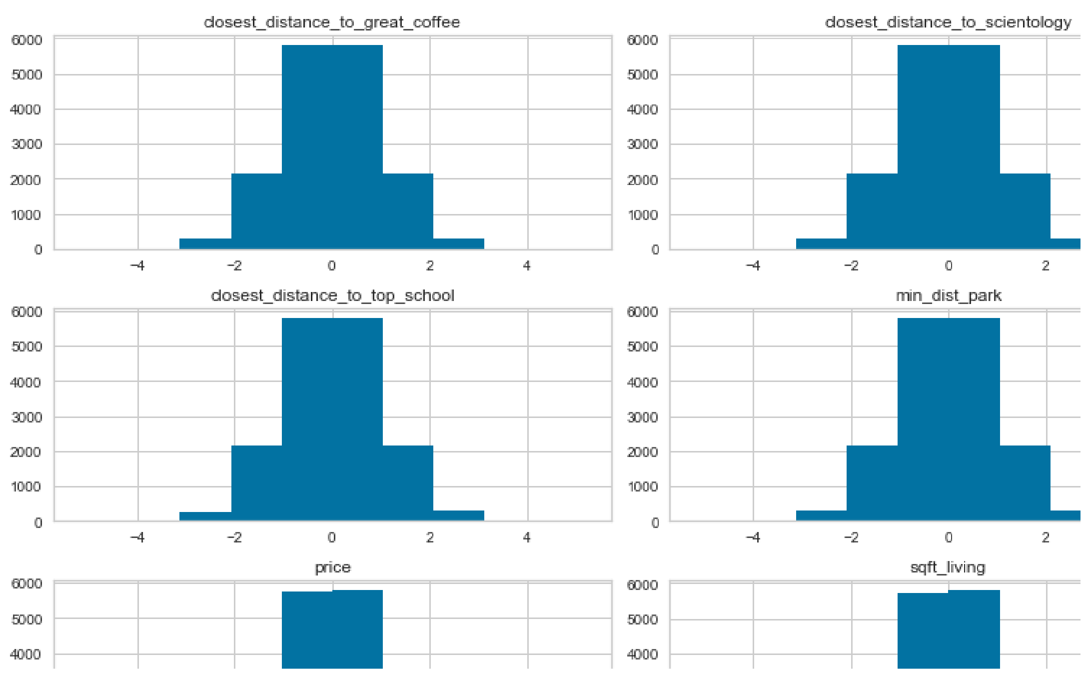
```
Skewness: 0.003628284074277272
Kurtosis: 0.04166152893410047
```



```
In [256]: plt.figure(figsize=(8,4))
sns.distplot(df['closest_distance_to_great_coffee'])
plt.title("Closest Distance to Great Coffee (Quant)", fontsize=2(
plt.xlabel('Closest Distance to Great Coffee');
print("Skewness:", df['closest_distance_to_great_coffee'].skew()
print("Kurtosis:", df['closest_distance_to_great_coffee'].kurt()
```

```
Skewness: -0.001082783246642902
Kurtosis: 0.020161693402130698
```


```
In [254]: df[['sqft_living', 'closest_distance_to_great_coffee', 'min_dist_
plt.tight_layout();
```




```
In [244]: grade_dums = pd.get_dummies(df.grade, prefix='grade', drop_first=
```

In [245]:
```
df = df.drop(['grade'], axis=1)
df = pd.concat([df, grade_dums], axis=1)
df.head()
```

Out[245]:

	price	sqft_living	lat	long	min_dist_park	closest_distance_to_top_school	clo
0	-1.60	-1.08	47.51	-122.26	-0.31	-1.61	
1	0.49	0.94	47.72	-122.32	0.92	-0.50	
2	-2.54	-2.14	47.74	-122.23	-0.84	0.36	
3	0.78	0.17	47.52	-122.39	-0.08	0.30	
4	0.37	-0.22	47.62	-122.05	0.02	0.08	

5 rows × 21 columns

In [246]:
```
df['interaction'] = df['closest_distance_to_top_school'] * df['c
features = ['sqft_living', 'closest_distance_to_great_coffee', 'r
target = ['price']
X = df[features]
y = df[target]

# running an iteration of the model with interaction column and i
X_train, X_test, y_train, y_test = train_test_split(X,y, random_s

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

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

```
R^2:  0.6308144610145117
```

By quantile tranforming our data to achieve a more normal distribution, we are able to a higher R2 score.

In [258]:
```
# saving copy of dataframe as csv file
#df.to_csv('./data/all_features_quant_transformed.csv')
```

Price Per Square Foot

While we were happy with the increasing R2 score, we wanted to experiment with a ne making a predictive model for price per square foot, as opposed to just price. By honir this target, our goal was to more accurately predict the value of a home based on our f

In [328]:
```
# reading the csv file
df = pd.read_csv('data/kc_house_data.csv')
# previewing the DataFrame
df.head()
```

Out[328]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floo
0	7129300520	10/13/2014	221900.00	3	1.00	1180	5650	1.(
1	6414100192	12/9/2014	538000.00	3	2.25	2570	7242	2.(
2	5631500400	2/25/2015	180000.00	2	1.00	770	10000	1.(
3	2487200875	12/9/2014	604000.00	4	3.00	1960	5000	1.(
4	1954400510	2/18/2015	510000.00	3	2.00	1680	8080	1.(

5 rows × 21 columns

In [329]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21597 entries, 0 to 21596
Data columns (total 21 columns):
id               21597 non-null int64
date             21597 non-null object
price            21597 non-null float64
bedrooms         21597 non-null int64
bathrooms        21597 non-null float64
sqft_living      21597 non-null int64
sqft_lot         21597 non-null int64
floors           21597 non-null float64
waterfront       19221 non-null float64
view             21534 non-null float64
condition        21597 non-null int64
grade            21597 non-null int64
sqft_above       21597 non-null int64
sqft_basement    21597 non-null object
yr_built         21597 non-null int64
yr_renovated     17755 non-null float64
zipcode          21597 non-null int64
lat              21597 non-null float64
long             21597 non-null float64
sqft_living15    21597 non-null int64
sqft_lot15       21597 non-null int64
dtypes: float64(8), int64(11), object(2)
memory usage: 3.5+ MB
```

In [330]:

```
# creating price per sqft column
df['price_per_sqft'] = (df['price'] / df['sqft_living'])
df.head()
```

Out[330]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floo
0	7129300520	10/13/2014	221900.00	3	1.00	1180	5650	1.(
1	6414100192	12/9/2014	538000.00	3	2.25	2570	7242	2.(
2	5631500400	2/25/2015	180000.00	2	1.00	770	10000	1.(
3	2487200875	12/9/2014	604000.00	4	3.00	1960	5000	1.(
4	1954400510	2/18/2015	510000.00	3	2.00	1680	8080	1.(

5 rows × 22 columns

In [331]:

```
df.price_per_sqft.describe()
```

Out[331]:

```
count   21597.00
mean      264.14
std       110.00
min        87.59
25%       182.29
50%       244.64
75%       318.33
max       810.14
Name: price_per_sqft, dtype: float64
```

In [334]:
```
plt.figure(figsize=(8,4))
sns.distplot(df['price_per_sqft'])
plt.title("Distribution of Price Per Sqft", fontsize=20)
plt.xlabel('Price Per Sqft');
print("Skewness:", df['price_per_sqft'].skew())
print("Kurtosis:", df['price_per_sqft'].kurt())
```

```
Skewness: 1.2469211620378835
Kurtosis: 2.0993152010383684
```

Narrowing down our data

We opted to use price per square foot as the factor by which to narrow our data. We re
outliers and focused on our main data by filtering for data within 1.5 standard deviatior
mean for price per square foot.

In [335]:
```
# finding the data that lies within 1.5 standard deviations from
std = df.price_per_sqft.std()
print('std: ',std)
mean = df.price_per_sqft.mean()
print('mean: ', mean)
std_1 = mean + std
std_1m = mean - std
std_15 = mean + (1.5*std)
std_15m = mean - (1.5*std)
print('mean +1 std: ',std_1)
print('mean -1 std: ',std_1m)
print('mean +1.5 std: ',std_15)
print('mean -1.5 std: ',std_15m)
```

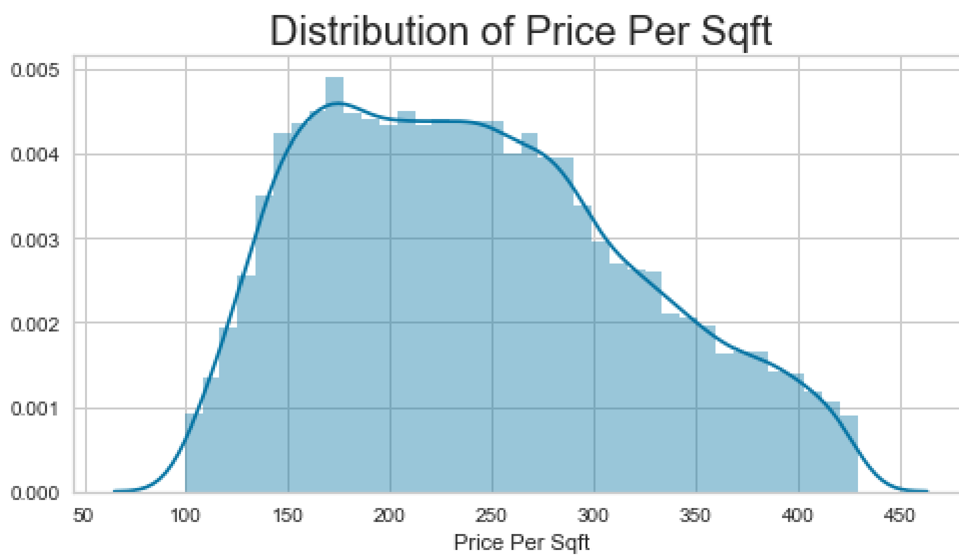
```
std:  110.00006067814525
mean:  264.1433683790251
mean +1 std:  374.14342905717035
mean -1 std:  154.14330770087986
mean +1.5 std:  429.143459396243
mean -1.5 std:  99.14327736180724
```

In [336]:
```
std = df.price_per_sqft.std()
mean = df.price_per_sqft.mean()
std_1 = mean + std
std_1m = mean - std
std_15 = mean + (1.5*std)
std_15m = mean - (1.5*std)
# removing outliers
# focusing on data within 1.5 standard deviations from the mean
df = df.loc[(df['price_per_sqft']<std_15) & (df['price_per_sqft'
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19785 entries, 0 to 21596
Data columns (total 22 columns):
id                  19785 non-null int64
date                19785 non-null object
price               19785 non-null float64
bedrooms            19785 non-null int64
bathrooms           19785 non-null float64
sqft_living         19785 non-null int64
sqft_lot            19785 non-null int64
floors              19785 non-null float64
waterfront          17586 non-null float64
view                19728 non-null float64
condition           19785 non-null int64
grade               19785 non-null int64
sqft_above          19785 non-null int64
sqft_basement       19785 non-null object
yr_built            19785 non-null int64
yr_renovated        16312 non-null float64
zipcode             19785 non-null int64
lat                 19785 non-null float64
long                19785 non-null float64
sqft_living15       19785 non-null int64
sqft_lot15          19785 non-null int64
price_per_sqft      19785 non-null float64
dtypes: float64(9), int64(11), object(2)
memory usage: 3.5+ MB
```

In [337]:
```
plt.figure(figsize=(8,4))
sns.distplot(df['price_per_sqft'])
plt.title("Distribution of Price Per Sqft", fontsize=20)
plt.xlabel('Price Per Sqft');
print("Skewness:", df['price_per_sqft'].skew())
print("Kurtosis:", df['price_per_sqft'].kurt())
```

```
Skewness: 0.37133132146882725
Kurtosis: -0.6982023030383484
```



In [338]:
```
#dropping unnecessary columns
drop = ['id','date', 'yr_built', 'bedrooms', 'bathrooms','sqft_lo
df = df.drop(columns = drop, axis=1)
```

In [339]:
```
df.columns
```

Out[339]:
```
Index(['price', 'sqft_living', 'grade', 'lat', 'long', 'price_pe
dtype='object')
```

In [340]:
```
df.isnull().sum()
```

Out[340]:
```
price             0
sqft_living       0
grade             0
lat               0
long              0
price_per_sqft    0
dtype: int64
```

Now that we had all new parameters, we needed to pull in the data again so that it was
outliers based on our new target variable, price per square foot.

King County Parks

In [341]:
```
# importing park data REVISED
# now including only parks (removing forests, natural areas, and
# for entire data scraping process, please see other notebook

# reading the csv file
king_parks = pd.read_csv('data/ParkAddresses_Revised_wLatLong.csv
# previewing the DataFrame
king_parks.head()
```

Out[341]:

	Name	Address	Combined	
ID				
0	NaN	NaN	NaN	r
1	NaN	NaN	NaN	r
2	NaN	NaN	NaN	r
3	Big Finn Hill Park	NE 138th and Juanita Drive NE Kirkland WA 98028	47.72417796430824, -122.2384511052857	47
4	NaN	NaN	NaN	r

In [342]:
```
king_parks.isnull().sum()
```

Out[342]:
```
Name        85
Address     85
Combined    85
Lat         85
Long        85
dtype: int64
```

In [343]:
```
king_parks.dropna(inplace=True)
```

In [344]:
```python
#create function to find distances between all points in DF and :
def find_distance(dataframe):
    """
    Calculates distance between points of interest and houses.

    Generates a distance matrix for distances between houses and
    Calculates distance from each point in dataframe (df) to poi
    Converts latitude and longitude to radians in order to calcu
    Returns values as kilometers.

    Parameters:
    dataframe (Pandas DataFrame object): user input name of Panda

    Returns:
    Matrix of distances.

    """
    dist = sklearn.neighbors.DistanceMetric.get_metric('haversine

    #convert lat and long to radians
    dataframe[['lat_radians','long_radians']] = (np.radians(data

    #create list matrix (results in km)
    dist_matrix = (dist.pairwise
    (df[['lat_radians_A','long_radians_A']],
     dataframe[['lat_radians','long_radians']])*6371)

    #return a matrix DataFrame
    return pd.DataFrame(dist_matrix)
```

In [345]:
```python
#convert lat and long to radians in housing data
df[['lat_radians_A','long_radians_A']] = (np.radians(df.loc[:,['
```

In [346]:
```python
park_matrix = find_distance(king_parks)
```

In [347]:
```python
#find min distance in each row
park_min_matrix = park_matrix.where(park_matrix.values == park_ma
    axis=1)[:,None]).drop_duplicates()
```

In [348]:
```python
#create a new column with only min distance and remove the rest
park_min_matrix['min_dist_park'] = park_min_matrix[park_min_matr
    lambda x: ','.join(x.dropna().astype(str)),
    axis=1)
nearest_park = park_min_matrix['min_dist_park']
```

In [349]:
```python
df = df.join(nearest_park)
```

In [350]:
```
df.head()
```

Out[350]:

	price	sqft_living	grade	lat	long	price_per_sqft	lat_radians_A	long_rad
0	221900.00	1180	7	47.51	-122.26	188.05	0.83	
1	538000.00	2570	7	47.72	-122.32	209.34	0.83	
2	180000.00	770	6	47.74	-122.23	233.77	0.83	
3	604000.00	1960	7	47.52	-122.39	308.16	0.83	
4	510000.00	1680	8	47.62	-122.05	303.57	0.83	

In [351]:
```
df['min_dist_park']= df['min_dist_park'].astype('float64')
```

King County Top Schools

In [352]:
```
# importing school data
# for entire data obtaining process, please see other notebook

# reading the csv file
top_schools_df = pd.read_csv('data/top_schools.csv')
# previewing the DataFrame
top_schools_df.head()
```

Out[352]:

	Unnamed: 0	year	ncessch	school_name	state_name	lea_name	zip_location
0	43	2015	530039000058	Ardmore Elementary School	Washington	Bellevue School District	98008
1	44	2015	530039000060	Bellevue High School	Washington	Bellevue School District	98004
2	45	2015	530039000062	Bennett Elementary School	Washington	Bellevue School District	98008
3	46	2015	530039000063	Cherry Crest Elementary School	Washington	Bellevue School District	98005
4	47	2015	530039000064	Chinook Middle School	Washington	Bellevue School District	98004

In [353]:
```
top_schools_df.drop(columns = 'Unnamed: 0', axis=1, inplace=True
```

In [354]:
```
top_schools_df.head()
```

Out[354]:

	year	ncessch	school_name	state_name	lea_name	zip_location	latitude	lo
0	2015	530039000058	Ardmore Elementary School	Washington	Bellevue School District	98008	47.64	
1	2015	530039000060	Bellevue High School	Washington	Bellevue School District	98004	47.60	
2	2015	530039000062	Bennett Elementary School	Washington	Bellevue School District	98008	47.62	
3	2015	530039000063	Cherry Crest Elementary School	Washington	Bellevue School District	98005	47.64	
4	2015	530039000064	Chinook Middle School	Washington	Bellevue School District	98004	47.63	

In [356]:
```
top_school_coordinates = []
x = round(top_schools_df.latitude, 2)
y = round(top_schools_df.longitude, 2)
top_school_coordinates = list(zip(x,y))
```

In [357]:
```
for i in range(len(top_school_coordinates)):
    df[f'top_school_{i}'] = distance_to(top_school_coordinates[i

top_school_cols = []
for i in range(len(top_school_coordinates)):
    top_school_cols.append(f'top_school_{i}')
    df['closest_distance_to_top_school'] = df[top_school_cols].m
```

In [358]:
```
df.drop(columns = top_school_cols, axis=1, inplace=True)
rad_cols = ['lat_radians_A', 'long_radians_A']
df.drop(columns=rad_cols, axis=1, inplace=True)
df.head()
```

Out[358]:

	price	sqft_living	grade	lat	long	price_per_sqft	min_dist_park	closest_
0	221900.00	1180	7	47.51	-122.26	188.05	2.04	
1	538000.00	2570	7	47.72	-122.32	209.34	5.67	
2	180000.00	770	6	47.74	-122.23	233.77	1.34	
3	604000.00	1960	7	47.52	-122.39	308.16	2.45	
4	510000.00	1680	8	47.62	-122.05	303.57	3.72	

King County Top 10 Coffee Shops

In [359]:

```
def get_keys(path):
    """Retrieves API key from files as api_key."""
    with open(path) as f:
        return json.load(f)

keys = get_keys("/Users/dtunnicliffe/.secret/yelp_api.json")
api_key = keys['api_key']

term = 'coffee'
location = 'King County, WA'
SEARCH_LIMIT = 10
espresso = pd.DataFrame([])
def yelp(term, location, SEARCH_LIMIT):
    """
    Creates a new dataframe of information retrieved from yelp AI

    Searches businesses and returns top results based on criteri
    Makes API call as if searching on yelp.
    Returns relevant information for businesses such as name, lo

    Parameters:
    term (str): user input term to search for.
    location (str): user input city, state, or zip code to searcl
    SEARCH_LIMIT (int): user input number of results to return.

    Returns:
    New dataframe populated with requested information.

    """
    global espresso
    url = 'https://api.yelp.com/v3/businesses/search'
    headers = {
    'Authorization': f'Bearer {api_key}',
    }
    url_params = {
    'term': term.replace(' ', '+'),
    'location': location.replace(' ', '+'),
    'limit': SEARCH_LIMIT,
    'sort_by': 'rating'
    }
    response = requests.get(url, headers=headers, params=url_para
    df_temp = pd.DataFrame.from_dict(response.json()['businesses
    espresso = espresso.append(df_temp)
    return espresso
```

In [360]:

```
espresso = yelp(term, location, SEARCH_LIMIT)
```

In [361]:

```
espresso.head()
```

Out[361]:

	id	alias	name	im
0	S6CXIQ5KrMpTPZf1eNMa2w	five-stones-coffee-company-redmond	Five Stones Coffee Company	htt media3.fl.yelpcdn.com/bphoto/Om
1	EWqgeiGor-aVJIMLc8iSKw	boon-boona-coffee-renton	Boon Boona Coffee	htt media3.fl.yelpcdn.com/bphoto/tV
2	v7xfqk9f7N8A98AQ2kddWg	anchorhead-coffee-bellevue-3	Anchorhead Coffee	htt media3.fl.yelpcdn.com/bphoto/Er
3	t2DOOFh-oJLddtpxbVlDrQ	huxdotter-coffee-north-bend	Huxdotter Coffee	htt media3.fl.yelpcdn.com/bphoto/M(
4	-MzbuOLr2kAoqlQY8w7ECA	pioneer-coffee-north-bend-north-bend	Pioneer Coffee - North Bend	htt media3.fl.yelpcdn.com/bphoto/5

In [362]:

```
great_coffee_coordinates = []
x = [round(coordinate['latitude'], 2) for coordinate in espresso
y = [round(coordinate['longitude'], 2) for coordinate in espress
great_coffee_coordinates = list(zip(x,y))
```

In [363]:

```
for i in range(len(great_coffee_coordinates)):
    df[f'great_coffee_{i}'] = distance_to(great_coffee_coordinat

great_coffee_cols = []
for i in range(len(great_coffee_coordinates)):
    great_coffee_cols.append(f'great_coffee_{i}')
    df['closest_distance_to_great_coffee'] = df[great_coffee_col
```

In [364]:

```
#dropping unnecessary columns
df = df.drop(columns = great_coffee_cols, axis=1)
df.head()
```

Out[364]:

	price	sqft_living	grade	lat	long	price_per_sqft	min_dist_park	closest_
0	221900.00	1180	7	47.51	-122.26	188.05	2.04	
1	538000.00	2570	7	47.72	-122.32	209.34	5.67	
2	180000.00	770	6	47.74	-122.23	233.77	1.34	
3	604000.00	1960	7	47.52	-122.39	308.16	2.45	
4	510000.00	1680	8	47.62	-122.05	303.57	3.72	

King County Churches of Scientology

In [365]:

```
#locations pulled from scientology-seattle.org
church_of_scientology_mission = (47.818100, -122.315430)
church_of_scientology_washington = (47.622380, -122.361020)
church_of_scientology_life_improvement_center = (47.615060, -122
```

In [367]:

```
#creating new columns of distances from houses to point of intere
df['distance_to_scientology_m'] = distance_to(church_of_scientolo
df['distance_to_scientology_w'] = distance_to(church_of_scientolo
df['distance_to_scientology_l'] = distance_to(church_of_scientolo
df['closest_distance_to_scientology'] = df[['distance_to_sciento]
                                            'distance
                                            'distance
```

In [368]:

```
sci_cols = ['distance_to_scientology_m', 'distance_to_scientology
            'distance_to_scientology_l']
df.drop(columns = sci_cols, axis=1, inplace=True)
```

In [369]:

```
df.head()
```

Out[369]:

	price	sqft_living	grade	lat	long	price_per_sqft	min_dist_park	closest_
0	221900.00	1180	7	47.51	-122.26	188.05	2.04	
1	538000.00	2570	7	47.72	-122.32	209.34	5.67	
2	180000.00	770	6	47.74	-122.23	233.77	1.34	
3	604000.00	1960	7	47.52	-122.39	308.16	2.45	
4	510000.00	1680	8	47.62	-122.05	303.57	3.72	

Quantile Transformation

In [370]:
```
df.isnull().sum()
```

Out[370]:
```
price                                0
sqft_living                          0
grade                                0
lat                                  0
long                                 0
price_per_sqft                       0
min_dist_park                     2290
closest_distance_to_top_school       0
closest_distance_to_great_coffee     0
closest_distance_to_scientology      0
dtype: int64
```

In [371]:
```
df.dropna(inplace=True)
```

In [373]:
```
df.corr()
```

Out[373]:

	price	sqft_living	grade	lat	long	price_per_sqft
price	1.00	0.76	0.71	0.37	0.07	0.52
sqft_living	0.76	1.00	0.76	0.08	0.22	-0.10
grade	0.71	0.76	1.00	0.11	0.22	0.12
lat	0.37	0.08	0.11	1.00	-0.10	0.54
long	0.07	0.22	0.22	-0.10	1.00	-0.18
price_per_sqft	0.52	-0.10	0.12	0.54	-0.18	1.00
min_dist_park	-0.01	-0.00	-0.01	-0.01	-0.01	-0.00
closest_distance_to_top_school	-0.35	-0.09	-0.10	-0.66	0.01	-0.50
closest_distance_to_great_coffee	-0.19	-0.15	-0.16	-0.16	-0.31	-0.09
closest_distance_to_scientology	-0.30	0.04	0.03	-0.72	0.63	-0.55

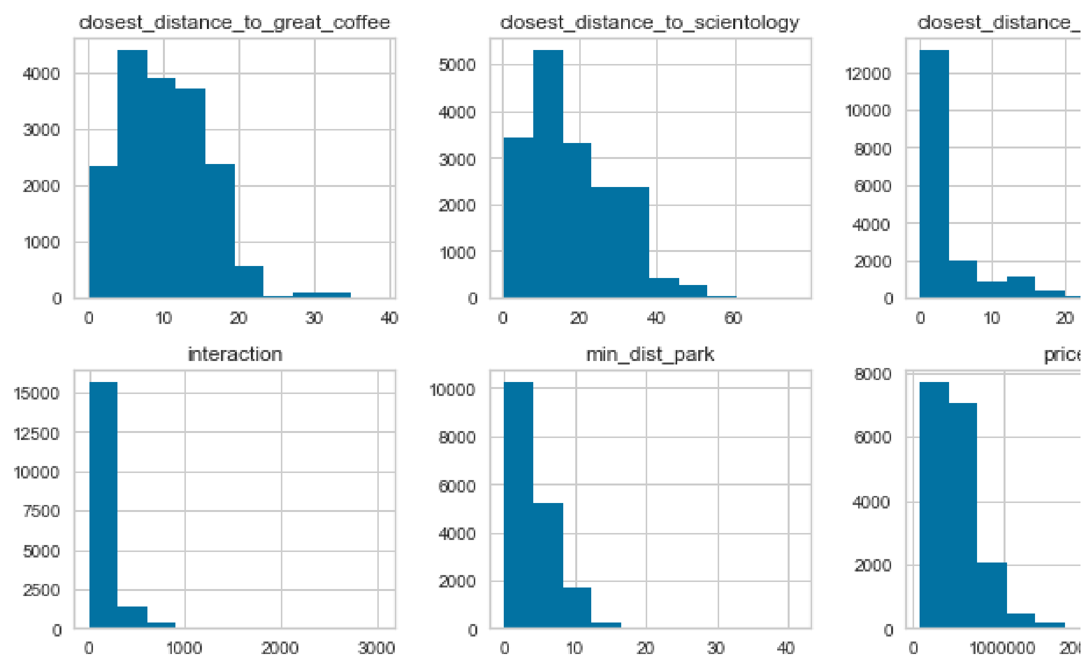
Since closest distance to top school and closest distance to scientology have multicoli
creating 'interaction' column to account for this relationship.

In [374]:
```
df['interaction'] = df['closest_distance_to_top_school'] * df['cl
df.head()
```

Out[374]:

grade	lat	long	price_per_sqft	min_dist_park	closest_distance_to_top_school	cl
7	47.51	-122.26	188.05	2.04	0.26	
7	47.72	-122.32	209.34	5.67	0.68	
6	47.74	-122.23	233.77	1.34	2.00	
7	47.52	-122.39	308.16	2.45	1.73	
8	47.62	-122.05	303.57	3.72	1.18	

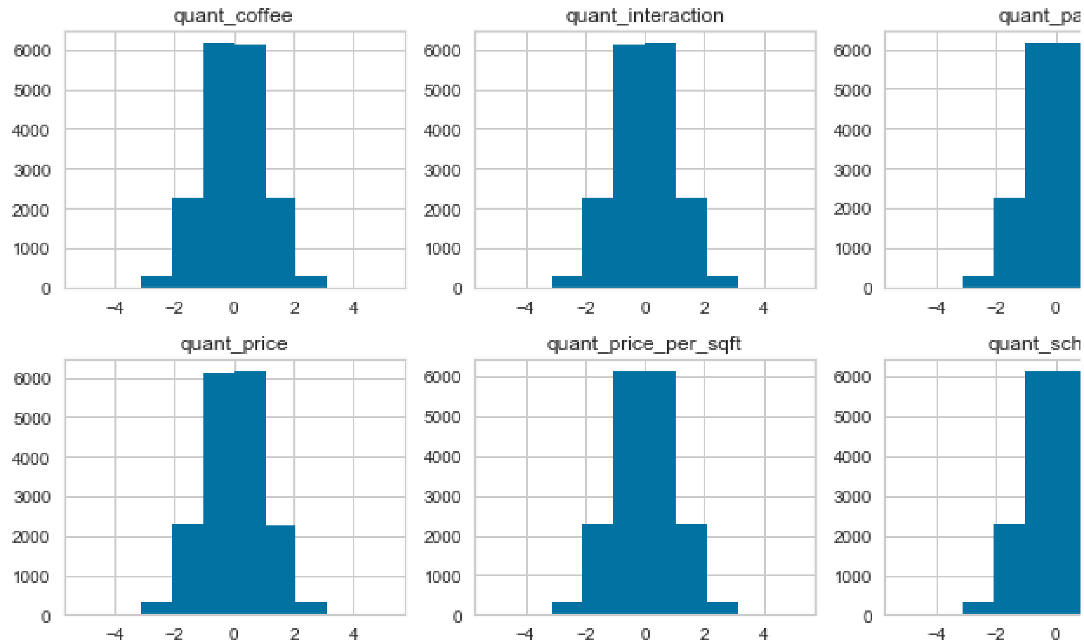
In [375]:
```
park', 'closest_distance_to_top_school', 'closest_distance_to_sci
```



Our features and target do not illustrate normal distrubtions.

In [378]:
```
# quantile-transforming our features and target
from sklearn.preprocessing import QuantileTransformer
qt = QuantileTransformer(output_distribution='normal')
df['quant_sqft_living'] = qt.fit_transform(df[['sqft_living']])
df['quant_coffee'] = qt.fit_transform(df[['closest_distance_to_gr
df['quant_parks'] = qt.fit_transform(df[['min_dist_park']])
df['quant_schools'] = qt.fit_transform(df[['closest_distance_to_t
df['quant_scientology'] = qt.fit_transform(df[['closest_distance_
df['quant_price'] = qt.fit_transform(df[['price']])
df['quant_price_per_sqft'] = qt.fit_transform(df[['price_per_sqft
df['quant_interaction'] = qt.fit_transform(df[['interaction']])
```

```
In [379]: quant_coffee', 'quant_parks', 'quant_schools', 'quant_scientology
```



Our quantile transformation led to a much more normal distribution for our features and

```
In [380]: grade_dums = pd.get_dummies(df.grade, prefix='grade', drop_first=
```

```
In [381]: df = df.drop(['grade'], axis=1)
          df = pd.concat([df, grade_dums], axis=1)
          df.head()
```

Out[381]:

listance_to_great_coffee	closest_distance_to_scientology	interaction	...	quant_interaction
4.39	12.71	3.33	...	-1.11
14.81	10.80	7.37	...	-0.50
10.63	10.84	21.71	...	0.08
14.48	11.55	19.97	...	0.05
8.55	21.18	24.98	...	0.16

```
In [383]: # saving copy of dataframe as csv file
          #df.to_csv('./data/all_features_ppsqft_quant.csv')
```