Price Prediction Models for Birkshire Hathaway Real Estate Group

Overview

This project analyzes the data from kc_house_data.csv and ZIP Codes. The kc_house_data.csv data consisted of detailed info of residential properties in King County, WA including sale price (prediction target). "ZIP Codes data consisted of all the ZIP Codes in King County with associated city.

We are using multiple linear regression modeling to analyze house sale prices and provide suggestions for Birkshire Hathaway Real Estate Group.

Business Problem

Washington's Birkshire Hathaway Real Estate Group are looking for efficient ways to refine and streamline the buying & selling process of residential properties in King County, WA for their clients. Our price prediction models aim to provide the company with estimated prices for each city and region based on the clients' preferences. Consequently, the company can focus more on the targeted area with a pricing reference.

Data Understanding

The King County House Data (kc_house_data.csv) has all the residential property details in King County, WA. We combined with ZIP Code data and organized the property details by each city. Each city data set varies in sizes and average price. So we also grouped the cities into 13 regions based on location proximity, average price, and similiar features. Since some of zipcodes are being shared with multiple towns, we found the center point of each city using latitude and longitude to group the nearby cities with similar prices.

In [34]:
```
#imported necessary packages

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as stats
import seaborn as sns
import statsmodels
from statsmodels.formula.api import ols
import warnings

warnings.filterwarnings("ignore")

plt.style.use('seaborn-darkgrid')
```

In [35]:
```
#Loaded in clean data (cleaned in student.ipynb)

df = pd.read_csv('data/testing_data.csv')
```

In [36]:
```
df.drop(columns=['Unnamed: 0'], inplace=True)
```

In [37]:
```
df.grade = df.grade.astype('category')
df.grade = df.grade.cat.reorder_categories(['3 Poor', '4 Low', '5 Fair', '6 Low Average
                                            '9 Better', '10 Very Good', '11 Excelle
df.condition = df.condition.astype('category')
df.condition = df.condition.cat.reorder_categories(['Fair', 'Poor', 'Average',
                                                    'Good', 'Very Good'])
df.view = df.view.astype('category')
df.view = df.view.cat.reorder_categories(['NONE', 'FAIR', 'AVERAGE', 'GOOD', 'EXCELLENT

df.view = df.view.cat.codes
df.grade = df.grade.cat.codes
df.condition =df.condition.cat.codes
```

In [38]:
```
#copied citites data mapped by zipcode

cities = {'Algona': [98001], 'Auburn': [98001, 98002, 98003, 98023, 98063, 98071, 98092
          'Beaux Arts Village': [98004], 'Bellevue':[98004, 98005, 98006, 98007, 98008,
           'Clyde Hill': [98004], 'Hunts Point': [98004], 'Yarrow Point': [98004], 'Blac
          'Bothell': [98011, 98041, 98028], 'Burton': [98013], 'Vashion': [98013], 'Carn
          'Duvall':[98019], 'Enumclaw': [98022], 'Fall City': [98024], 'Hobart': [98025]
          'Issaquah': [98027], 'Kent': [98030, 98031, 98032, 98035, 98042, 98064], 'Kirk
          'Maple Valley': [98038], 'Medina': [98039], 'Mercer Island': [98040], 'Kenmore
          'Covington':[98042], 'North Bend': [98045], 'Pacific':[98047], 'Preston':[9805
          'Ravensdale':[98051], 'Redmond': [98052, 98053, 98073, 98074], 'Redondo': [980
          'Newcastle': [98056, 98059,], 'Seahurst':[98062], 'Snoqualmie': [98065, 98068]
          'Vashon': [98070], 'Woodinville': [98072], 'Sammamish':[98075, 98075], 'Issaqu
          'Seattle': [98101, 98102, 98103, 98104, 98105, 98106, 98107, 98108, 98109, 981
                      98126, 98131, 98132, 98133, 98134, 98136, 98138, 98144, 98145, 981
           'Tukwila': [98108, 98138, 98168, 98178, 98188],
          'Shoreline': [98133, 98155, 98177 ], 'Burien': [98146, 98148, 98166, 98168 ],
           'Lake Forest Park': [98155, 98155, 98155], 'Baring': [98224], 'Skykomish': [9

#created pandas dataframes for each city with the columns in df

data = {}

for i in cities.keys():
    if i in df.columns:
        data[i] = df[df[i] == 1]
    else:
        continue
```

center_geolcation(geolocations) function was taken from amites github
https://gist.github.com/amites/3718961

In [39]:
```
from math import cos, sin, atan2, sqrt, pi

def center_geolocation(geolocations):
    """
    Provide a relatively accurate center lat, lon returned as a list pair, given
    a list of list pairs.
    ex: in: geolocations = ((lat1,lon1), (lat2,lon2),)
        out: (center_lat, center_lon)
```

```
    """
    x = 0
    y = 0
    z = 0

    for lat, lon in geolocations:
        lat = float(lat *(pi/180))
        lon = float(lon * (pi/180))
        x += cos(lat) * cos(lon)
        y += cos(lat) * sin(lon)
        z += sin(lat)

    x = float(x / len(geolocations))
    y = float(y / len(geolocations))
    z = float(z / len(geolocations))

    degrees1 = atan2(z, sqrt(x * x + y * y))
    degrees2 = atan2(y, x)

    return (degrees1 * (180/pi), degrees2 *(180/pi))
```

In [40]:
```
#creating a dictonary of centerpoints of each city

coordinates = {}

for i in data.keys():
    x = data[i]['lat']
    y = data[i]['long']
    pair = list(zip(x,y))

    coordinates[i] = pair

center_location = {}

for i in data.keys():
    center_location[i] = center_geolocation(coordinates[i])

center_location.pop('Clyde Hill')
center_location.pop('Hunts Point')
center_location.pop('Yarrow Point')
```

Out[40]:
```
(47.616183826606125, -122.20518721213313)
```

In [41]:
```
#mapped the cities for each city

import folium

lat = 47.613417665161194
long = -122.33245505039801

#Create a map of the area
base_map = folium.Map([lat, long], zoom_start=13)

for p in center_location.keys():
    lat = center_location[p][0]
    long = center_location[p][1]
    marker = folium.Marker(location=[lat, long])
    marker.add_to(base_map)
    popup_text = "City: {}, Latitude: {}, Longitude: {}".format(p,lat,long)
```

```
    popup = folium.Popup(popup_text, parse_html=True)
    marker = folium.Marker(location=[lat, long], popup=popup)
    marker.add_to(base_map)
base_map
```

Out[41]: Make this Notebook Trusted to load map: File -> Trust Notebook

In [42]:
```
#created a dictionary of dataframes by region using the above map

regions_df = {}

regions_df['Southwest'] = df[(df['Federal Way'] == 1) | (df['Auburn'] == 1) | (df['Algo
regions_df['Southeast'] = df[(df['Enumclaw'] == 1) | (df['Black Diamond'] == 1) | (df['
                             (df['Covington'] == 1) | (df['Kent'] == 1)]
regions_df['Islands'] = df[df['Vashon'] == 1]

regions_df['South_of_Greater_Seattle'] = df[(df['Des Moines'] == 1) | (df['Normandy Par
                             (df['Burien'] == 1) | (df['Tukwila'] == 1)]
regions_df['Seattle_Region'] = df[df['Seattle'] == 1]
regions_df['Southeast_of_Greater_Seattle'] = df[(df['Renton'] == 1) | (df['Newcastle']
regions_df['Rich'] = df[(df['Mercer Island'] == 1) | (df['Bellevue'] == 1) | (df['Beaux
                             (df['Medina'] == 1)]
regions_df['Kirkland Region'] = df[df['Kirkland'] == 1]
regions_df['NorthEast'] = df[(df['Kenmore'] == 1) | (df['Bothell'] == 1) | (df['Woodinv
regions_df['NorthWest'] = df[(df['Shoreline'] == 1) | (df['Lake Forest Park'] == 1)]
regions_df['Redmond Region'] = df[df['Redmond'] == 1]
regions_df['Suburban'] = df[(df['Sammamish'] == 1) | (df['Issaquah'] == 1)]
regions_df['Rural'] = df[(df['Duvall'] == 1) | (df['Carnation'] == 1) | (df['Fall City'
                             (df['Snoqualmie'] == 1) | (df['North Bend'] == 1)]
```

Exploratory Data Anayslis

Data Exploration by City

In [43]:
```
city_dfs = {}
```

```
for i in cities.keys():
    if i in df.columns:
        data = df[df[i] == 1]
        city_dfs[i] = data
    else:
        continue

city_dfs.pop('Beaux Arts Village')
city_dfs.pop('Clyde Hill')
city_dfs.pop('Hunts Point')
```

Out[43]:

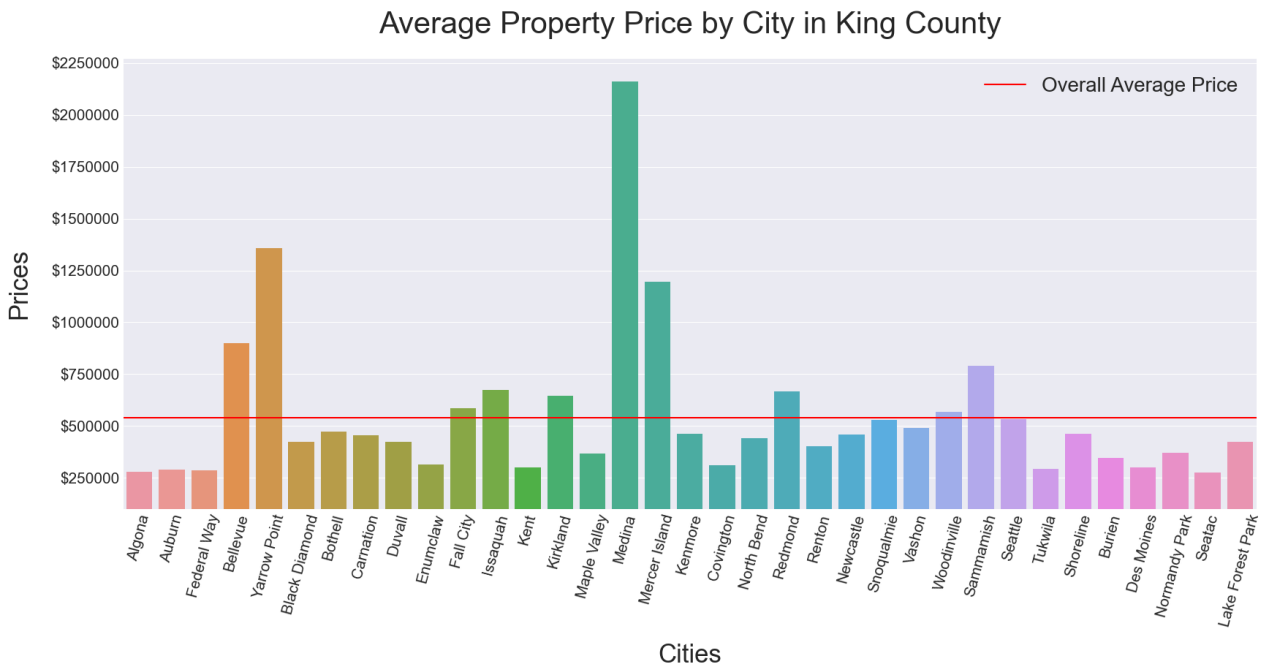
	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grad
66	975000.0	4	2.50	2720	11049	2.0	0	0	2	
69	1330000.0	5	2.25	3200	20158	1.0	0	0	2	
102	1090000.0	3	2.50	2920	8113	2.0	0	0	2	
123	1450000.0	4	2.75	2750	17789	1.5	0	0	2	
265	2900000.0	4	3.25	5050	20100	1.5	0	2	2	
...	...	...	...	...	...	...	...	...	...	
21259	1750000.0	4	2.75	3560	8975	2.0	0	0	2	
21316	3000000.0	4	3.75	5090	14823	1.0	0	0	2	
21319	999999.0	3	2.50	2100	4097	2.0	0	0	2	
21354	1700000.0	4	3.50	3830	8963	2.0	0	0	2	
21386	1540000.0	5	3.75	4470	8088	2.0	0	0	2	

317 rows × 60 columns

In [44]:

```
mean_prices = {}

for key, values in city_dfs.items():
    mean = city_dfs[key]['price'].mean()
    mean_prices[key] = mean

fig, ax = plt.subplots(figsize=(20,8))

sns.barplot(x=list(mean_prices.keys()), y = list(mean_prices.values()))
ax.set_xticks(list(range(len(city_dfs.keys()))))
ax.set_xticklabels(city_dfs.keys(), rotation =75, fontsize=15)
ax.set_yticks(list(range(0, 2500000, 250000)))
ax.set_yticklabels(['${}'.format(i) for i in range(0,2500000, 250000)], fontsize=15)

ax.set_ylabel('Prices', fontsize= 25)
ax.yaxis.labelpad = 20


ax.set_xlabel('Cities', fontsize= 25)
ax.xaxis.labelpad = 15
```

```
ax.set_ylim(100000)
ax.axhline(y = df.price.mean(), color = 'r', linestyle = '-', label='Overall Average Pr

ax.set_title('Average Property Price by City in King County', fontsize=30, pad=25)
plt.legend(fontsize=20)



plt.show()
```

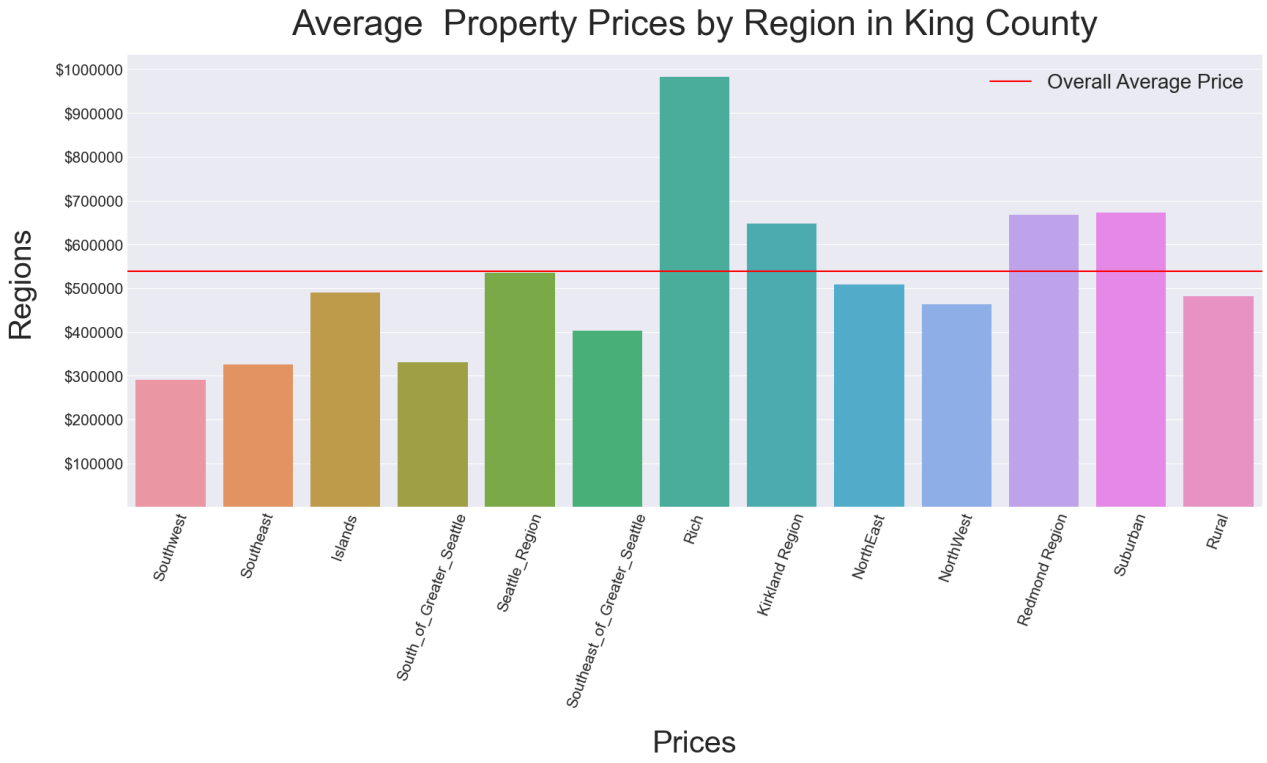


Data Exploration by Region

In [45]:

```
mean_prices = {}

for key, values in regions_df.items():
    mean = regions_df[key]['price'].mean()
    mean_prices[key] = mean

fig, ax = plt.subplots(figsize=(20,8))

sns.barplot(x=list(mean_prices.keys()), y = list(mean_prices.values()))

ax.set_xticks(list(range(len(mean_prices))))
ax.set_xticklabels(regions_df.keys(), rotation=70, fontsize=15)
ax.set_yticks(list(range(100000, 1100000, 100000)))
ax.set_yticklabels(['${}'.format(i) for i in range(100000, 1100000, 100000)], fontsize=

ax.set_xlabel('Prices', fontsize=30)
ax.set_ylabel('Regions', fontsize=30)

ax.yaxis.labelpad = 20
ax.xaxis.labelpad =20

ax.set_title('Average  Property Prices by Region in King County', fontsize =35, pad=20)
```

```
ax.axhline(y = df.price.mean(), color = 'r', linestyle = '-', label = 'Overall Average

plt.legend(fontsize=20)
plt.show()
```



Linear Regression Models

In this section, we looped through different models in order to determine the best model for each city and model. We realized our r-squared scores were most affect by the technique we used to clean our data and the amount of features in the model. Therefore, we partitoned our code for linear regression into 2 sections, with three subsections each.

Linear Regression Models for each Region

In [46]:

```
#imported necessary packages
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error

from sklearn.preprocessing import PolynomialFeatures

#poly is used for interactions between highly coorelated ind. variables
poly = PolynomialFeatures(include_bias=False, interaction_only=True)

#variables we looped through
col_selector = ['price', 'bedrooms', 'bathrooms', 'sqft_living', 'grade', 'view', 'wate
                'has_basement', 'condition', 'floors', 'recently_renovated']


all_models = {'df_type':[], 'cleaning_method': [], 'ind_vars': [], 'rsquared': [], 'MAE
```

```
                'num_of_feats': [], 'model': [], 'test_size': []}

def add_model(df_type, cleaning_method, ind_vars, rsquared, MAE, RMSE, test_size, num_o

    """
    This function takes in the following paramaters and adds them to a dictonary. The p
    is to later change the dictonary into a pandas dataframe to find the best linear re
    for each region by it's best r-squared value


    """

    all_models['df_type'].append(df_type)
    all_models['cleaning_method'].append(cleaning_method)
    all_models['ind_vars'].append(ind_vars)
    all_models['rsquared'].append(rsquared)
    all_models['MAE'].append(MAE)
    all_models['RMSE'].append(RMSE)
    all_models['num_of_feats'].append(num_of_feats)
    all_models['model'].append(model)
    all_models['test_size'].append(test_size)
```

Not Cleaning Outliers - Regions

We did not clean outliers before doing linear regression on each region.

In [47]:

```
cleaning_method = 'None'

for counter in range(2,6):

    for i in regions_df.keys():

        df_type = i

        #chooses the highest coorelated variables to price
        col_lst = list(regions_df[i].loc[:, col_selector].corr().price.sort_values(asce

        x = regions_df[i][col_lst]
        X = poly.fit_transform(x)
        y = regions_df[i]['price']


        test = None

        #we use a higher test size for smaller datasets
        if len(regions_df[i]) < 300:
            test = 0.3
            X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test)
        else:
            test = 0.2
            X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test)

        lr = LinearRegression()

        lr.fit(X_train, y_train)
        rsquared = lr.score(X_train, y_train)
        y_pred = lr.predict(X_test)
```

```
        MAE = mean_absolute_error(y_pred, y_test)
        RMSE = np.sqrt(mean_squared_error(y_pred, y_test))

        feats = lr.n_features_in_

        model_dict= {}
        error_dict = {}

        #as mentioned before this function is adding all the features of the model to a
        add_model(df_type, cleaning_method, col_lst, rsquared, MAE, RMSE, test, feats,
```

Cleaning outliers by quantile range

In [48]:
```
#creates a copy of the regions_df dictionary
quantile_dfs = regions_df.copy()

for i in quantile_dfs.keys():
    for x in col_selector:

        data = quantile_dfs[i][x]

        q1 = data.quantile(0.25)
        q3 = data.quantile(0.75)
        iqr = q3 - q1
        lower = q1 - 1.5*iqr
        upper = q3 + 1.5*iqr

        #replaces outliers with quantile range
        quantile_dfs[i][x] = np.where(data > upper, upper, data)
        quantile_dfs[i][x] = np.where(data < lower, lower, data)
```

In [49]:
```
cleaning_method = 'quantile'

for counter in range(2,6):

    for i in quantile_dfs.keys():

        df_type = i

        col_lst = list(quantile_dfs[i].loc[:, col_selector].corr().price.sort_values(as

        x = quantile_dfs[i][col_lst]
        X = poly.fit_transform(x)
        y = quantile_dfs[i]['price']


        test = None

        if len(quantile_dfs[i]) < 300:
            test = 0.3
            X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test)
        else:
            test = 0.2
            X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test)

        lr = LinearRegression()

        lr.fit(X_train, y_train)
```

```
        rsquared = lr.score(X_train, y_train)
        y_pred = lr.predict(X_test)


        MAE = mean_absolute_error(y_pred, y_test)
        RMSE = np.sqrt(mean_squared_error(y_pred, y_test))

        feats = lr.n_features_in_

        add_model(df_type, cleaning_method, col_lst, rsquared, MAE, RMSE, test, feats,
```

Cleaning model by logarithmic

In [50]:
```
#makes a copy of the regions_df dict
log_dfs = regions_df.copy()

#Logs each coorelated variable
for i in log_dfs.keys():
    for x in col_selector:
        log_dfs[i][x] = log_dfs[i][x].map(lambda k: np.log(k) if k > 0 else 0)
```

In [51]:
```
cleaning_method = 'log'

for counter in range(2,6):

    for i in regions_df.keys():

        df_type = i

        col_lst = list(regions_df[i].loc[:, col_selector].corr().price.sort_values(asce

        x = regions_df[i][col_lst]
        X = poly.fit_transform(x)
        y = regions_df[i]['price']


        test = None

        if len(regions_df[i]) < 300:
            test = 0.3
            X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test)
        else:
            test = 0.2
            X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test)

        lr = LinearRegression()

        lr.fit(X_train, y_train)
        rsquared = lr.score(X_train, y_train)
        y_pred = lr.predict(X_test)


        MAE = mean_absolute_error(np.exp(y_pred), np.exp(y_test))
        RMSE = np.sqrt(mean_squared_error(np.exp(y_pred), np.exp(y_test)))

        feats = lr.n_features_in_

        add_model(df_type, cleaning_method, col_lst, rsquared, MAE, RMSE, test, feats,
```

Linear Models for Cities

No Cleaning Outliers

In [52]:
```
cleaning_method = 'None'

for counter in range(2,6):

    for i in city_dfs.keys():

        df_type = i

        col_lst = list(city_dfs[i].loc[:, col_selector].corr().price.sort_values(ascend

        x = city_dfs[i][col_lst]
        X = poly.fit_transform(x)
        y = city_dfs[i]['price']


        test = None

        if len(city_dfs[i]) < 300:
            test = 0.4
            X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test)
        else:
            test = 0.3
            X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test)

        lr = LinearRegression()

        lr.fit(X_train, y_train)
        rsquared = lr.score(X_train, y_train)
        y_pred = lr.predict(X_test)


        MAE = mean_absolute_error(y_pred, y_test)
        RMSE = np.sqrt(mean_squared_error(y_pred, y_test))

        feats = lr.n_features_in_

        add_model(df_type, cleaning_method, col_lst, rsquared, MAE, RMSE, test, feats,
```

Cleaning outliers in city set with quantile replacement

In [53]:
```
quantile_city_dfs = city_dfs.copy()

for i in quantile_city_dfs.keys():
    for x in col_selector:

        data = quantile_city_dfs[i][x]

        q1 = data.quantile(0.25)
        q3 = data.quantile(0.75)
        iqr = q3 - q1
        lower = q1 - 1.5*iqr
```

```
        upper = q3 + 1.5*iqr
        quantile_city_dfs[i][x] = np.where(data > upper, upper, data)
        quantile_city_dfs[i][x] = np.where(data < lower, lower, data)
```

In [54]:
```
cleaning_method = 'quantile'

for counter in range(2,6):

    for i in quantile_city_dfs.keys():

        df_type = i

        col_lst = list(quantile_city_dfs[i].loc[:, col_selector].corr().price.sort_valu

        x = quantile_city_dfs[i][col_lst]
        X = poly.fit_transform(x)
        y = quantile_city_dfs[i]['price']


        test = None

        if len(quantile_city_dfs[i]) < 300:
            test = 0.3
            X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test)
        else:
            test = 0.2
            X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test)

        lr = LinearRegression()

        lr.fit(X_train, y_train)
        rsquared = lr.score(X_train, y_train)
        y_pred = lr.predict(X_test)


        MAE = mean_absolute_error(y_pred, y_test)
        RMSE = np.sqrt(mean_squared_error(y_pred, y_test))

        feats = lr.n_features_in_

        add_model(df_type, cleaning_method, col_lst, rsquared, MAE, RMSE, test, feats,
```

Cleaning Outliers in city set with logartimethic

In [55]:
```
log_city_dfs = city_dfs.copy()

for i in log_city_dfs.keys():
    for x in col_selector:
        log_city_dfs[i][x] = log_city_dfs[i][x].map(lambda k: np.log(k) if k > 0 else 0
```

In [56]:
```
cleaning_method = 'log'

for counter in range(2,6):

    for i in log_city_dfs.keys():

        df_type = i
```

```
        col_lst = list(log_city_dfs[i].loc[:, col_selector].corr().price.sort_values(as

        x = log_city_dfs[i][col_lst]
        X = poly.fit_transform(x)
        y = log_city_dfs[i]['price']


        test = None

        if len(log_city_dfs[i]) < 300:
            test = 0.3
            X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test)
        else:
            test = 0.2
            X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test)

        lr = LinearRegression()

        lr.fit(X_train, y_train)
        rsquared = lr.score(X_train, y_train)
        y_pred = lr.predict(X_test)


        MAE = mean_absolute_error(np.exp(y_pred), np.exp(y_test))
        RMSE = np.sqrt(mean_squared_error(np.exp(y_pred), np.exp(y_test)))

        feats = lr.n_features_in_

        add_model(df_type, cleaning_method, col_lst, rsquared, MAE, RMSE, test, feats,
```

Evaluating all Models

In this section, we look at all our models and the determine the best model for each city and region by the highest r-squared value.

In [57]:
```
#created a dataframe using the all_models dict that has been updated with values with e
final_df = pd.DataFrame(data=all_models)
```

In [58]:
```
final_df
```

Out[58]:

	df_type	cleaning_method	ind_vars	rsquared	MAE	RMSE	num
0	Southwest	None	[sqft_living]	0.672155	41661.614388	67762.323819	
1	Southeast	None	[sqft_living]	0.586859	49592.584515	74634.233136	
2	Islands	None	[grade]	0.437831	105928.983011	145931.683213	
3	South_of_Greater_Seattle	None	[sqft_living]	0.454190	88659.765826	154529.280405	
4	Seattle_Region	None	[sqft_living]	0.525146	165110.572783	237147.579792	
...	...	...	...	...	...	...	...
571	Burien	log	[sqft_living, grade, bathrooms, view]	0.680716	55016.741076	70202.376150	

	df_type	cleaning_method	ind_vars	rsquared	MAE	RMSE	nun
572	Des Moines	log	[sqft_living, bathrooms, view, grade]	0.627544	38243.281389	51498.043671	
573	Normandy Park	log	[sqft_living, grade, bathrooms, view]	0.694191	61909.107927	83917.668253	
574	Seatac	log	[sqft_living, bathrooms, grade, bedrooms]	0.566547	38078.125238	52577.049215	
575	Lake Forest Park	log	[sqft_living, grade, bathrooms, bedrooms]	0.682916	50584.984622	67097.571495	

576 rows × 9 columns

As you see above, we have 576 linear regression models

Separating Models by Max R-Squared value

In [59]:
```
#created 4 dictonaries

#sep_dfs will contain individual dataframes filered by df_type-- ex: Southwest region w
sep_dfs = {}

#max_rsquares will have the values of the linear regresssion models corresponding to df
max_rsquares = {}

#contains the indexes where the rsquared value is maxed -- Ex: {'Southwest': [4]}
indexes = {}

#contains the models by highest r-squared model for each df_type -- Ex: {'Southwest': L
best_models = {}

for i in final_df.df_type.unique():
    sep_dfs[i] = final_df[final_df.df_type == i]

for i in sep_dfs.keys():
    max_rsquares[i] = sep_dfs[i].rsquared.max()

for key, value in max_rsquares.items():
    indexes[key] = sep_dfs[key].index[sep_dfs[key].rsquared == value].tolist()

for i in sep_dfs.keys():
    best_models[i] = sep_dfs[i].loc[indexes[i][0]]
```

In [60]:
```
best_models
```

Out[60]:
```
{'Southwest': df_type                                      Southwest
```

```
cleaning_method                                        None
ind_vars           [sqft_living, grade, bathrooms, view]
rsquared                                           0.749948
MAE                                                 39232.9
RMSE                                                68846.4
num_of_feats                                             10
model                                  LinearRegression()
test_size                                               0.2
Name: 39, dtype: object,
'Southeast': df_type                                          Southeast
cleaning_method                                          None
ind_vars           [sqft_living, grade, bathrooms, floors]
rsquared                                           0.675884
MAE                                                 43696.3
RMSE                                                68295.2
num_of_feats                                             10
model                                    LinearRegression()
test_size                                               0.2
Name: 40, dtype: object,
'Islands': df_type                                            Islands
cleaning_method                                      quantile
ind_vars           [grade, sqft_living, bathrooms, floors]
rsquared                                            0.66333
MAE                                                  106902
RMSE                                                 132288
num_of_feats                                             10
model                                    LinearRegression()
test_size                                               0.3
Name: 93, dtype: object,
'South_of_Greater_Seattle': df_type                           South_of_Greater_Seattle
cleaning_method                                        None
ind_vars           [sqft_living, grade, view, waterfront]
rsquared                                           0.761513
MAE                                                 65213.1
RMSE                                                 102658
num_of_feats                                             10
model                                  LinearRegression()
test_size                                               0.2
Name: 42, dtype: object,
'Seattle_Region': df_type                                     Seattle_Region
cleaning_method                                        None
ind_vars           [sqft_living, grade, bathrooms, view]
rsquared                                           0.679454
MAE                                                  133318
RMSE                                                 187332
num_of_feats                                             10
model                                  LinearRegression()
test_size                                               0.2
Name: 43, dtype: object,
'Southeast_of_Greater_Seattle': df_type               Southeast_of_Greater_Seattle
cleaning_method                                    None
ind_vars           [sqft_living, grade, bathrooms]
rsquared                                     0.780638
MAE                                           68668.1
RMSE                                           151631
num_of_feats                                        6
model                            LinearRegression()
test_size                                         0.2
Name: 31, dtype: object,
'Rich': df_type                                               Rich
cleaning_method                                        None
ind_vars           [sqft_living, grade, bathrooms, view]
rsquared                                           0.707056
MAE                                                  249277
RMSE                                                 369354
```

```
num_of_feats                                        10
model                                LinearRegression()
test_size                                          0.2
Name: 45, dtype: object,
'Kirkland Region': df_type                                    Kirkland Region
cleaning_method                                    None
ind_vars          [sqft_living, grade, bathrooms, view]
rsquared                                       0.815735
MAE                                              117258
RMSE                                             179590
num_of_feats                                        10
model                                LinearRegression()
test_size                                          0.2
Name: 46, dtype: object,
'NorthEast': df_type                                     NorthEast
cleaning_method                                  log
ind_vars          [sqft_living, grade, bathrooms]
rsquared                                    0.737649
MAE                                          57083.2
RMSE                                         83472.2
num_of_feats                                       6
model                          LinearRegression()
test_size                                        0.2
Name: 138, dtype: object,
'NorthWest': df_type                                          NorthWest
cleaning_method                                    None
ind_vars          [sqft_living, grade, view, bathrooms]
rsquared                                       0.82563
MAE                                            74283.6
RMSE                                            124364
num_of_feats                                        10
model                                LinearRegression()
test_size                                          0.2
Name: 48, dtype: object,
'Redmond Region': df_type                                               Redmond Region
cleaning_method                                       None
ind_vars          [sqft_living, grade, bathrooms, waterfront]
rsquared                                          0.815653
MAE                                                  67051
RMSE                                                 100976
num_of_feats                                             10
model                                   LinearRegression()
test_size                                              0.2
Name: 49, dtype: object,
'Suburban': df_type                                               Suburban
cleaning_method                                       None
ind_vars          [sqft_living, grade, bathrooms, waterfront]
rsquared                                           0.85087
MAE                                                76127.6
RMSE                                                112685
num_of_feats                                            10
model                                   LinearRegression()
test_size                                              0.2
Name: 50, dtype: object,
'Rural': df_type                                          Rural
cleaning_method                                    None
ind_vars          [sqft_living, grade, bathrooms]
rsquared                                    0.808381
MAE                                          59731.8
RMSE                                          106473
num_of_feats                                       6
model                          LinearRegression()
test_size                                        0.2
Name: 38, dtype: object,
'Algona': df_type                                           Algona
```

```
cleaning_method                                          None
ind_vars          [sqft_living, grade, bathrooms, floors]
rsquared                                             0.77685
MAE                                                  32939.8
RMSE                                                 47801.4
num_of_feats                                              10
model                                  LinearRegression()
test_size                                                0.3
Name: 261, dtype: object,
'Auburn': df_type                                                Auburn
cleaning_method                                          None
ind_vars          [sqft_living, grade, bathrooms, view]
rsquared                                             0.74991
MAE                                                  36968.9
RMSE                                                 55149.3
num_of_feats                                              10
model                                  LinearRegression()
test_size                                                0.3
Name: 262, dtype: object,
'Federal Way': df_type                                           Federal Way
cleaning_method                                          None
ind_vars          [sqft_living, grade, bathrooms, view]
rsquared                                             0.750777
MAE                                                   37684.2
RMSE                                                  52795.5
num_of_feats                                              10
model                                  LinearRegression()
test_size                                                0.3
Name: 263, dtype: object,
'Bellevue': df_type                                              Bellevue
cleaning_method                                       quantile
ind_vars          [sqft_living, grade, bathrooms, view]
rsquared                                             0.673018
MAE                                                    176173
RMSE                                                   233482
num_of_feats                                              10
model                                  LinearRegression()
test_size                                                0.2
Name: 404, dtype: object,
'Yarrow Point': df_type                                          Yarrow Point
cleaning_method                                       quantile
ind_vars          [sqft_living, grade, bathrooms, view]
rsquared                                             0.811119
MAE                                                    213156
RMSE                                                   274923
num_of_feats                                              10
model                                  LinearRegression()
test_size                                                0.2
Name: 405, dtype: object,
'Black Diamond': df_type                                   Black Diamond
cleaning_method                                     None
ind_vars          [sqft_living, grade, bathrooms]
rsquared                                    0.800103
MAE                                          96536.9
RMSE                                          138082
num_of_feats                                       6
model                           LinearRegression()
test_size                                      0.4
Name: 231, dtype: object,
'Bothell': df_type                                                     Bothell
cleaning_method                                       quantile
ind_vars          [sqft_living, grade, bathrooms, bedrooms]
rsquared                                             0.732204
MAE                                                   49793.3
RMSE                                                  64012.5
```

```
num_of_feats                                          10
model                                 LinearRegression()
test_size                                            0.2
Name: 407, dtype: object,
'Carnation': df_type                             Carnation
cleaning_method                                   None
ind_vars          [sqft_living, bathrooms, grade]
rsquared                                      0.843486
MAE                                            86601.5
RMSE                                            118678
num_of_feats                                         6
model                           LinearRegression()
test_size                                          0.4
Name: 233, dtype: object,
'Duvall': df_type                                   Duvall
cleaning_method                                   None
ind_vars          [sqft_living, grade, bathrooms, bedrooms]
rsquared                                      0.796401
MAE                                            64247.9
RMSE                                           92743.2
num_of_feats                                        10
model                                 LinearRegression()
test_size                                          0.4
Name: 269, dtype: object,
'Enumclaw': df_type                               Enumclaw
cleaning_method                                     log
ind_vars          [sqft_living, grade, bathrooms, view]
rsquared                                      0.723268
MAE                                            55797.3
RMSE                                           76150.4
num_of_feats                                        10
model                                LinearRegression()
test_size                                          0.3
Name: 550, dtype: object,
'Fall City': df_type                             Fall City
cleaning_method                                   None
ind_vars          [sqft_living, grade, bathrooms, floors]
rsquared                                      0.917995
MAE                                             158920
RMSE                                            264418
num_of_feats                                        10
model                                 LinearRegression()
test_size                                          0.4
Name: 271, dtype: object,
'Issaquah': df_type                               Issaquah
cleaning_method                                   None
ind_vars          [sqft_living, grade, bathrooms, waterfront]
rsquared                                      0.844967
MAE                                            72187.5
RMSE                                            105960
num_of_feats                                        10
model                                 LinearRegression()
test_size                                          0.3
Name: 272, dtype: object,
'Kent': df_type                                   Kent
cleaning_method                                quantile
ind_vars          [sqft_living, grade, bathrooms]
rsquared                                       0.71103
MAE                                            34540.1
RMSE                                           49485.6
num_of_feats                                         6
model                           LinearRegression()
test_size                                          0.2
Name: 378, dtype: object,
'Kirkland': df_type                               Kirkland
```

```
cleaning_method                                        None
ind_vars           [sqft_living, grade, bathrooms, view]
rsquared                                           0.844456
MAE                                                  133287
RMSE                                                 214688
num_of_feats                                             10
model                                  LinearRegression()
test_size                                               0.3
Name: 274, dtype: object,
'Maple Valley': df_type                             Maple Valley
cleaning_method                                    None
ind_vars           [sqft_living, grade, bathrooms]
rsquared                                       0.765908
MAE                                             49055.6
RMSE                                            70093.2
num_of_feats                                          6
model                              LinearRegression()
test_size                                         0.3
Name: 240, dtype: object,
'Medina': df_type                                                  Medina
cleaning_method                                              None
ind_vars           [sqft_living, grade, bathrooms, bedrooms]
rsquared                                                 0.96843
MAE                                                       397894
RMSE                                                      526674
num_of_feats                                                  10
model                                       LinearRegression()
test_size                                                    0.4
Name: 276, dtype: object,
'Mercer Island': df_type                                  Mercer Island
cleaning_method                                             log
ind_vars           [sqft_living, grade, bathrooms, view]
rsquared                                           0.785802
MAE                                                  198591
RMSE                                                 268200
num_of_feats                                             10
model                                  LinearRegression()
test_size                                               0.3
Name: 557, dtype: object,
'Kenmore': df_type                                             Kenmore
cleaning_method                                          None
ind_vars           [sqft_living, grade, bathrooms, view]
rsquared                                           0.763873
MAE                                                 59331.4
RMSE                                                94924.4
num_of_feats                                             10
model                                  LinearRegression()
test_size                                               0.4
Name: 278, dtype: object,
'Covington': df_type                                         Covington
cleaning_method                                              None
ind_vars           [sqft_living, grade, bathrooms, floors]
rsquared                                           0.709883
MAE                                                 43908.6
RMSE                                                  70742
num_of_feats                                             10
model                                    LinearRegression()
test_size                                               0.3
Name: 279, dtype: object,
'North Bend': df_type                               North Bend
cleaning_method                                    None
ind_vars           [sqft_living, grade, bathrooms]
rsquared                                       0.890892
MAE                                             46810.8
RMSE                                            80559.3
```

```
num_of_feats                                      6
model                           LinearRegression()
test_size                                       0.4
Name: 245, dtype: object,
'Redmond': df_type                                                  Redmond
cleaning_method                                                        None
ind_vars           [sqft_living, grade, bathrooms, waterfront]
rsquared                                                           0.834163
MAE                                                                 69148.8
RMSE                                                                99431.9
num_of_feats                                                             10
model                                          LinearRegression()
test_size                                                               0.3
Name: 281, dtype: object,
'Renton': df_type                                             Renton
cleaning_method                                                 None
ind_vars           [sqft_living, grade, bathrooms]
rsquared                                         0.788575
MAE                                               67851.5
RMSE                                               127137
num_of_feats                                            6
model                                LinearRegression()
test_size                                           0.3
Name: 247, dtype: object,
'Newcastle': df_type                                                Newcastle
cleaning_method                                                      quantile
ind_vars           [grade, sqft_living, bathrooms, floors]
rsquared                                                           0.833286
MAE                                                                 55653.4
RMSE                                                                77178.9
num_of_feats                                                             10
model                                       LinearRegression()
test_size                                                               0.2
Name: 423, dtype: object,
'Snoqualmie': df_type                                                 Snoqualmie
cleaning_method                                                      quantile
ind_vars           [sqft_living, grade, bathrooms, bedrooms]
rsquared                                                           0.924002
MAE                                                                   42152
RMSE                                                                64241.8
num_of_feats                                                             10
model                                         LinearRegression()
test_size                                                               0.2
Name: 424, dtype: object,
'Vashon': df_type                                                   Vashon
cleaning_method                                                      quantile
ind_vars           [grade, sqft_living, bathrooms, floors]
rsquared                                                           0.695981
MAE                                                                  113601
RMSE                                                                 146360
num_of_feats                                                             10
model                                       LinearRegression()
test_size                                                               0.3
Name: 425, dtype: object,
'Woodinville': df_type                                              Woodinville
cleaning_method                                                        None
ind_vars           [sqft_living, grade, bathrooms, floors]
rsquared                                                           0.817661
MAE                                                                 80894.6
RMSE                                                                 125657
num_of_feats                                                             10
model                                       LinearRegression()
test_size                                                               0.4
Name: 286, dtype: object,
'Sammamish': df_type                                                Sammamish
```

```
cleaning_method                                        None
ind_vars           [waterfront, sqft_living, view, grade]
rsquared                                            0.87208
MAE                                                   78458
RMSE                                                 129156
num_of_feats                                             10
model                                  LinearRegression()
test_size                                               0.3
Name: 287, dtype: object,
'Seattle': df_type                                                Seattle
cleaning_method                                        None
ind_vars           [sqft_living, grade, bathrooms, view]
rsquared                                            0.68207
MAE                                                  134340
RMSE                                                 196467
num_of_feats                                             10
model                                LinearRegression()
test_size                                               0.3
Name: 288, dtype: object,
'Tukwila': df_type                                                Tukwila
cleaning_method                                        None
ind_vars           [sqft_living, grade, bathrooms, view]
rsquared                                           0.608507
MAE                                                 58702.2
RMSE                                                 104290
num_of_feats                                             10
model                                LinearRegression()
test_size                                               0.3
Name: 289, dtype: object,
'Shoreline': df_type                                              Shoreline
cleaning_method                                        None
ind_vars           [sqft_living, grade, view, bathrooms]
rsquared                                           0.813552
MAE                                                 77783.6
RMSE                                                 142798
num_of_feats                                             10
model                                LinearRegression()
test_size                                               0.3
Name: 290, dtype: object,
'Burien': df_type                                                 Burien
cleaning_method                                        None
ind_vars           [grade, view, sqft_living, bathrooms]
rsquared                                           0.813232
MAE                                                 62579.1
RMSE                                                86986.7
num_of_feats                                             10
model                                LinearRegression()
test_size                                               0.3
Name: 291, dtype: object,
'Des Moines': df_type                                             Des Moines
cleaning_method                                        None
ind_vars           [sqft_living, view, grade, bathrooms]
rsquared                                           0.809108
MAE                                                 55114.5
RMSE                                                77137.4
num_of_feats                                             10
model                                LinearRegression()
test_size                                               0.3
Name: 292, dtype: object,
'Normandy Park': df_type                                       Normandy Park
cleaning_method                                        None
ind_vars           [sqft_living, grade, view, waterfront]
rsquared                                           0.819583
MAE                                                 74757.4
RMSE                                                 111084
```

```
num_of_feats                                          10
model                                 LinearRegression()
test_size                                            0.3
Name: 293, dtype: object,
'Seatac': df_type                                 Seatac
cleaning_method                                     None
ind_vars           [sqft_living, grade, view, bathrooms]
rsquared                                         0.750822
MAE                                               42494.2
RMSE                                              63235.5
num_of_feats                                          10
model                                 LinearRegression()
test_size                                            0.3
Name: 294, dtype: object,
'Lake Forest Park': df_type                  Lake Forest Park
cleaning_method                                     None
ind_vars           [waterfront, sqft_living, view, grade]
rsquared                                         0.880827
MAE                                               70702.5
RMSE                                               179673
num_of_feats                                          10
model                                 LinearRegression()
test_size                                            0.3
Name: 295, dtype: object}
```

Model Prediction

In this section, we created functions to predict the prices based on selected user input. For example, if someone wants a property with 3 bedrooms, 2bathrooms and 1500 square feet. The output will be a dictonary of the predicted prices by each city and region

In [61]:
```
def total_model(dct, df_type, cleaning_method, ind_vars, rsquared, MAE, RMSE, test_size

    dct['df_type'].append(df_type)
    dct['cleaning_method'].append(cleaning_method)
    dct['ind_vars'].append(ind_vars)
    dct['rsquared'].append(rsquared)
    dct['MAE'].append(MAE)
    dct['RMSE'].append(RMSE)
    dct['num_of_feats'].append(num_of_feats)
    dct['model'].append(model)
    dct['test_size'].append(test_size)
```

In [62]:
```
def cities_prices(dct, cols):
    """

    This function takes in two paramaters: a dictornary and inputted columns. The dicti
    as the function loops through each model. The columns is the selected variables the
    Ex:

    dct = {'df_type':[], 'cleaning_method': [], 'ind_vars': [], 'rsquared': [], 'MAE':
           'num_of_feats': [], 'model': [], 'test_size': []}
    cols = ['bedrooms', 'grade', 'sqft_living']

    """

    poly = PolynomialFeatures(include_bias=False, interaction_only=True)
```

```
cleaning_method = ['None', 'quantile']

for clean in cleaning_method:

    if clean == 'quantile':

        quantile = city_dfs.copy()

        for key in quantile.keys():
            for col in col_selector:

                data = quantile[key][col]

                q1 = data.quantile(0.25)
                q3 = data.quantile(0.75)
                iqr = q3 - q1
                lower = q1 - 1.5*iqr
                upper = q3 + 1.5*iqr
                quantile[key][col] = np.where(data > upper, upper, data)
                quantile[key][col] = np.where(data < lower, lower, data)


        for i in quantile.keys():

            df_type = i

            x = quantile[i][cols]
            X = poly.fit_transform(x)
            y = quantile[i]['price']

            test = None

            if len(quantile[i]) < 300:
                test = 0.4
                X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
            else:
                test = 0.3
                X_train, X_test, y_train, y_test = train_test_split(X, y, test_size

            lr = LinearRegression()

            lr.fit(X_train, y_train)
            rsquared = lr.score(X_train, y_train)
            y_pred = lr.predict(X_test)


            MAE = mean_absolute_error(y_pred, y_test)
            RMSE = np.sqrt(mean_squared_error(y_pred, y_test))

            feats = lr.n_features_in_

            total_model(dct, df_type, clean, cols, rsquared, MAE, RMSE, test, feats


    elif clean == 'log':

        log = city_dfs.copy()

        for i in log.keys():
            for x in col_selector:
```

```
                log[i][x] = log[i][x].map(lambda k: np.log(k) if k > 0 else 0)


        for i in log.keys():

            df_type = i

            x = log[i][cols]
            X = poly.fit_transform(x)
            y = log[i]['price']


            test = None

            if len(log[i]) < 300:
                test = 0.4
                X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
            else:
                test = 0.3
                X_train, X_test, y_train, y_test = train_test_split(X, y, test_size

            lr = LinearRegression()

            lr.fit(X_train, y_train)
            rsquared = lr.score(X_train, y_train)
            y_pred = lr.predict(X_test)


            MAE = mean_absolute_error(np.exp(y_pred), np.exp(y_test))
            RMSE = np.sqrt(mean_squared_error(np.exp(y_pred), np.exp(y_test)))

            feats = lr.n_features_in_

            total_model(dct, df_type, clean, cols, rsquared, MAE, RMSE, test, feats

    else:

        norm = city_dfs.copy()

        for i in norm.keys():

            df_type = i

            x = norm[i][cols]
            X = poly.fit_transform(x)
            y = norm[i]['price']


            test = None

            if len(norm[i]) < 300:
                test = 0.4
                X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
            else:
                test = 0.3
                X_train, X_test, y_train, y_test = train_test_split(X, y, test_size

            lr = LinearRegression()

            lr.fit(X_train, y_train)
```

```
                rsquared = lr.score(X_train, y_train)
                y_pred = lr.predict(X_test)


                MAE = mean_absolute_error(y_pred, y_test)
                RMSE = np.sqrt(mean_squared_error(y_pred, y_test))

                feats = lr.n_features_in_

                total_model(dct, df_type, clean, cols, rsquared, MAE, RMSE, test, feats
```

In [63]:

```
def region_prices(dct, cols):

    """

    This function takes in two paramaters: a dictornary and inputted columns. The dicti
    as the function loops through each model. The columns is the selected variables the
    Ex:
    dct = {'df_type':[], 'cleaning_method': [], 'ind_vars': [], 'rsquared': [], 'MAE':
            'num_of_feats': [], 'model': [], 'test_size': []}
    cols = ['bedrooms', 'grade', 'sqft_living']

    """


    poly = PolynomialFeatures(include_bias=False, interaction_only=True)

    cleaning_method = ['None', 'quantile']

    for clean in cleaning_method:

        if clean == 'quantile':

            quantile = regions_df.copy()

            for key in quantile.keys():
                for col in col_selector:

                    data = quantile[key][col]

                    q1 = data.quantile(0.25)
                    q3 = data.quantile(0.75)
                    iqr = q3 - q1
                    lower = q1 - 1.5*iqr
                    upper = q3 + 1.5*iqr
                    quantile[key][col] = np.where(data > upper, upper, data)
                    quantile[key][col] = np.where(data < lower, lower, data)


            for i in quantile.keys():

                df_type = i

                x = quantile[i][cols]
                X = poly.fit_transform(x)
                y = quantile[i]['price']
```

```
            test = None

            if len(quantile[i]) < 300:
                test = 0.4
                X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
            else:
                test = 0.3
                X_train, X_test, y_train, y_test = train_test_split(X, y, test_size

            lr = LinearRegression()

            lr.fit(X_train, y_train)
            rsquared = lr.score(X_train, y_train)
            y_pred = lr.predict(X_test)


            MAE = mean_absolute_error(y_pred, y_test)
            RMSE = np.sqrt(mean_squared_error(y_pred, y_test))

            feats = lr.n_features_in_

            total_model(dct, df_type, clean, cols, rsquared, MAE, RMSE, test, feats


    elif clean == 'log':

        log = regions_df.copy()

        for i in log.keys():
            for x in col_selector:
                log[i][x] = log[i][x].map(lambda k: np.log(k) if k > 0 else 0)


        for i in log.keys():

            df_type = i

            x = log[i][cols]
            X = poly.fit_transform(x)
            y = log[i]['price']


            test = None

            if len(log[i]) < 300:
                test = 0.4
                X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
            else:
                test = 0.3
                X_train, X_test, y_train, y_test = train_test_split(X, y, test_size

            lr = LinearRegression()

            lr.fit(X_train, y_train)
            rsquared = lr.score(X_train, y_train)
            y_pred = lr.predict(X_test)


            MAE = mean_absolute_error(np.exp(y_pred), np.exp(y_test))
```

```
            RMSE = np.sqrt(mean_squared_error(np.exp(y_pred), np.exp(y_test)))

            feats = lr.n_features_in_

            total_model(dct, df_type, clean, cols, rsquared, MAE, RMSE, test, feats

        else:

            norm = regions_df.copy()

            for i in norm.keys():

                df_type = i

                x = norm[i][cols]
                X = poly.fit_transform(x)
                y = norm[i]['price']


                test = None

                if len(norm[i]) < 300:
                    test = 0.4
                    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
                else:
                    test = 0.3
                    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size

                lr = LinearRegression()

                lr.fit(X_train, y_train)
                rsquared = lr.score(X_train, y_train)
                y_pred = lr.predict(X_test)


                MAE = mean_absolute_error(y_pred, y_test)
                RMSE = np.sqrt(mean_squared_error(y_pred, y_test))

                feats = lr.n_features_in_

                total_model(dct, df_type, clean, cols, rsquared, MAE, RMSE, test, feats
```

In [66]:

```
def find_price(dct1, dct2, answers):

    """

    This function combines the cities_prices() and region_prices() model to predict the
    It takes in three dicts. dct1 and dct2 are the empty dicts that will be filled with
    the user wants. Ex:

    answers = {'bathrooms': 3, 'bedrooms': 4, 'sqft_living': 3000}

    dct1 = {'df_type':[], 'cleaning_method': [], 'ind_vars': [], 'rsquared': [], 'MAE':
            'num_of_feats': [], 'model': [], 'test_size': []}
    dct2 = {'df_type':[], 'cleaning_method': [], 'ind_vars': [], 'rsquared': [], 'MAE':
            'num_of_feats': [], 'model': [], 'test_size': []}
```

```
    """


    cols = list(answers.keys())
    vals1 = poly.fit_transform(pd.Series(answers).values.reshape(1,-1))

    cities_prices(dct1, cols)
    region_prices(dct2, cols)

    df1 = pd.DataFrame(data=dct1)
    df2 = pd.DataFrame(data=dct2)

    best_city_prices = {}
    best_region_prices = {}

    for i in city_dfs.keys():
        d = df1[df1.df_type == i]
        max_rsquared = d.rsquared.max()
        r = d.index[d.rsquared == max_rsquared].tolist()
        m = d.loc[r[0]]
        best_city_prices[i] = m.model.predict(vals1)

    for i in regions_df.keys():
        d = df2[df2.df_type == i]
        max_rsquared = d.rsquared.max()
        r = d.index[d.rsquared == max_rsquared].tolist()
        m = d.loc[r[0]]
        best_region_prices[i] = m.model.predict(vals1)


    min_city = min(best_city_prices, key=best_city_prices.get)
    min_val = best_city_prices[min_city]

    min_region = min(best_region_prices, key=best_region_prices.get)
    min_val_region = best_region_prices[min_region]

    max_city = max(best_city_prices, key=best_city_prices.get)
    max_val = best_city_prices[max_city]

    max_region = max(best_region_prices, key=best_region_prices.get)
    max_val_region = best_region_prices[max_region]


    return df1, df2, best_city_prices, best_region_prices
```

Testing the Product

In this last part, we test the find_price() function by testing three different sets of parameters.

In [67]:

```
test = {'bathrooms': 3, 'bedrooms': 4, 'sqft_living': 3000}

townz = {'df_type':[], 'cleaning_method': [], 'ind_vars': [], 'rsquared': [], 'MAE': []
            'num_of_feats': [], 'model': [], 'test_size': []}
areaz = {'df_type':[], 'cleaning_method': [], 'ind_vars': [], 'rsquared': [], 'MAE': []
            'num_of_feats': [], 'model': [], 'test_size': []}

frame, frame1, price, price1 = find_price(townz, areaz, test)
```

```
print("-------------------------------------------------")


test2 = {'bathrooms': 1, 'bedrooms': 1, 'sqft_living': 800}

town20 = {'df_type':[], 'cleaning_method': [], 'ind_vars': [], 'rsquared': [], 'MAE': [
            'num_of_feats': [], 'model': [], 'test_size': []}
area20 = {'df_type':[], 'cleaning_method': [], 'ind_vars': [], 'rsquared': [], 'MAE': [
            'num_of_feats': [], 'model': [], 'test_size': []}

frame2, frame22, price2, price22 = find_price(town20, area20, test2)

print("-------------------------------------------------")

test3 = {'bathrooms': 3, 'bedrooms': 6, 'sqft_living': 5000, 'grade':5 }

town300 = {'df_type':[], 'cleaning_method': [], 'ind_vars': [], 'rsquared': [], 'MAE':
            'num_of_feats': [], 'model': [], 'test_size': []}
area300 = {'df_type':[], 'cleaning_method': [], 'ind_vars': [], 'rsquared': [], 'MAE':
            'num_of_feats': [], 'model': [], 'test_size': []}

frame3, frame33, price3, price33 = find_price(town300, area300, test3)
```

```
-------------------------------------------------
-------------------------------------------------
```

Conclusions

This model is very large and complex based on the number of variales that we test. Our next step is to incorporate more data to train the model. So that we can find balance between increasing the R-squared and decreasing the error.

In [ ]: