US HOME PRICE INDEX PREDICTION

The data set contains the following parameters of 20 years that will affect HOME PRICE INDEX or HPI in future,

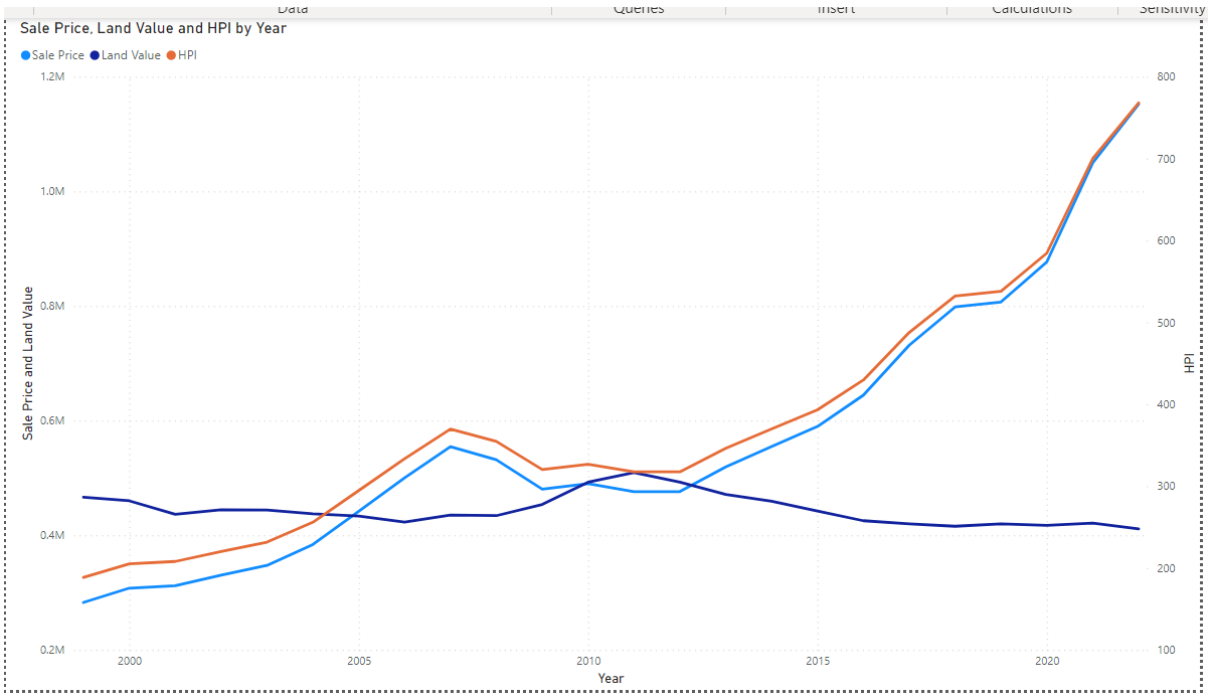
- Sale Price: Increase in sale price over the years can make changes in HPI.
- City : Some cities have high land values than others due to various reasons that affects HPI.
- Land value: Increase in land value over the years can make changes in HPL.
- Square feet(sqft): More demand in sqft means more land value that affects HPI.
- Stories: If a home have more stories then the price of home will be more compared to other homes.
- Beds: More beds mean more property value.
- Bath full: Full bath can add up to extra 20 charge on home price compared to half bath or quatre
- Bath 3 quatre
- Bath half
- Garb square feet
- Gara Square feet
- WFNT
- Golf
- Green Belt
- Noise traffic
- View rainer
- View Olympic

- View Cascades
- View territorial
- Submarket: If a home situated near a branded submarket, then the home price will be more compared to others.

All the variable from WFNT to View territorial are the extra benefits asked by the buyer, adding this can increase the cost of home to some extent that would affect HPI.

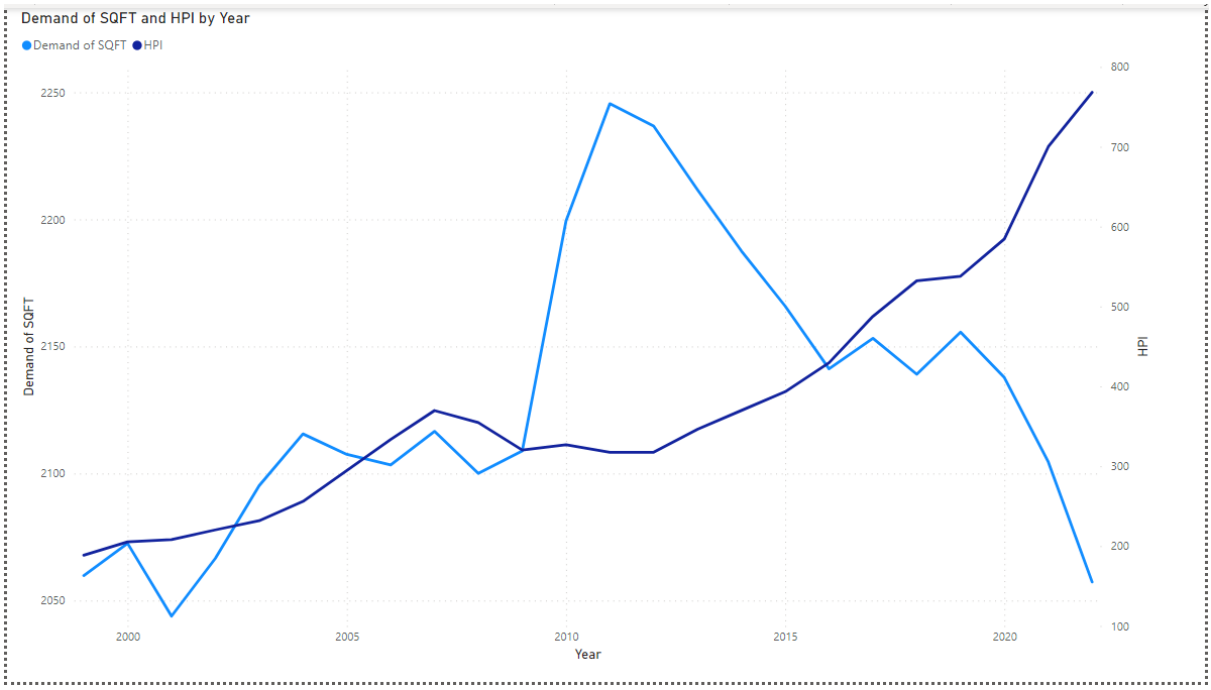
EXPLORATARY DATA ANALYSIS

Teck Stack Used: Power BI for visualization and graphs.

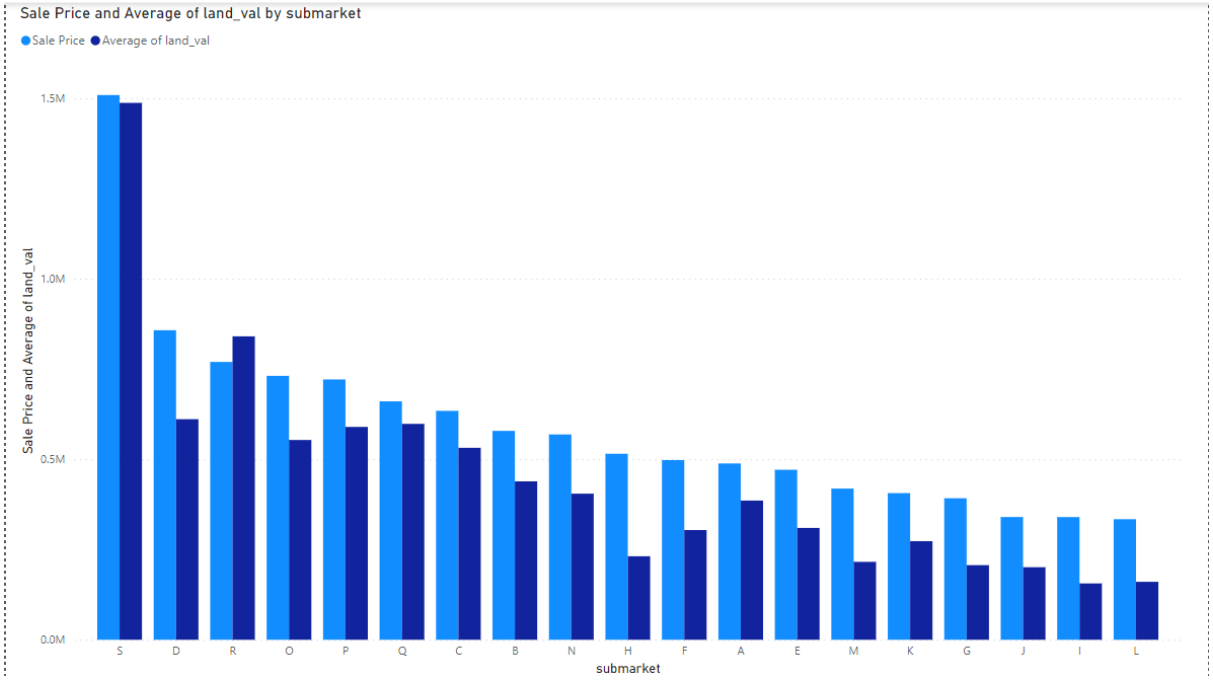


From this above graph we can observe that both sale price and HPI is positively correlated.

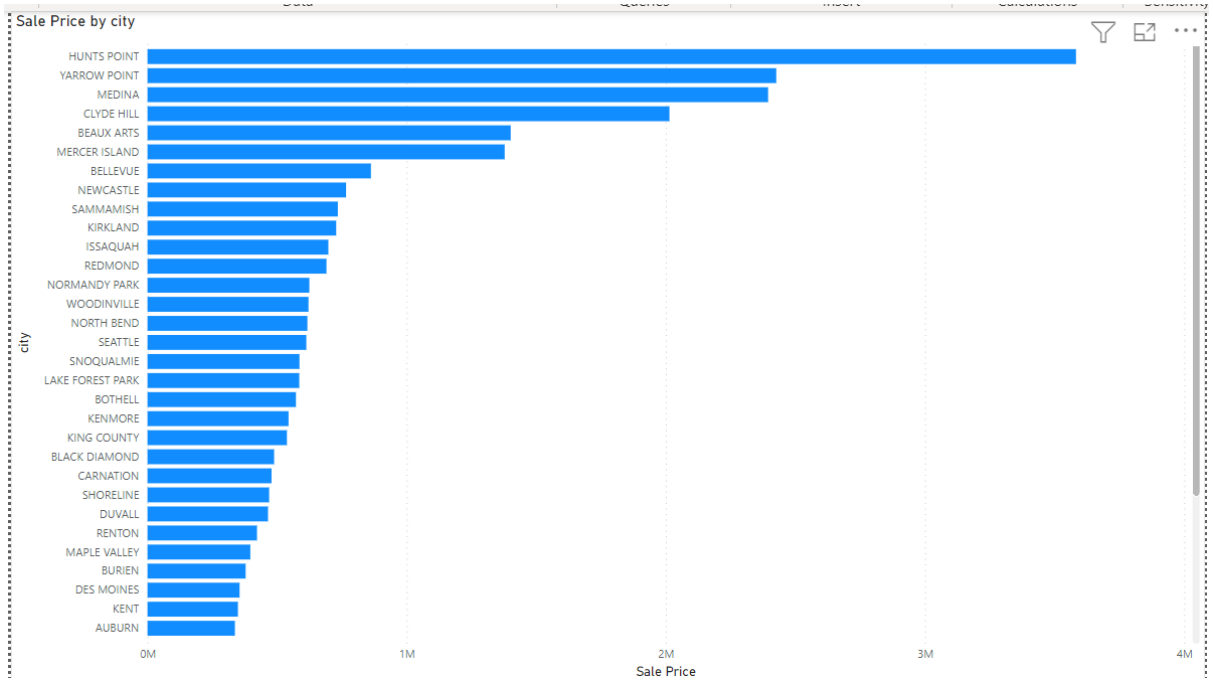
As the sale price increase HPI also increases. Although land value falls after 2010 yet HPI and land value increases. Hence, decreasing in land value does not affects sale price and HPI negatively.

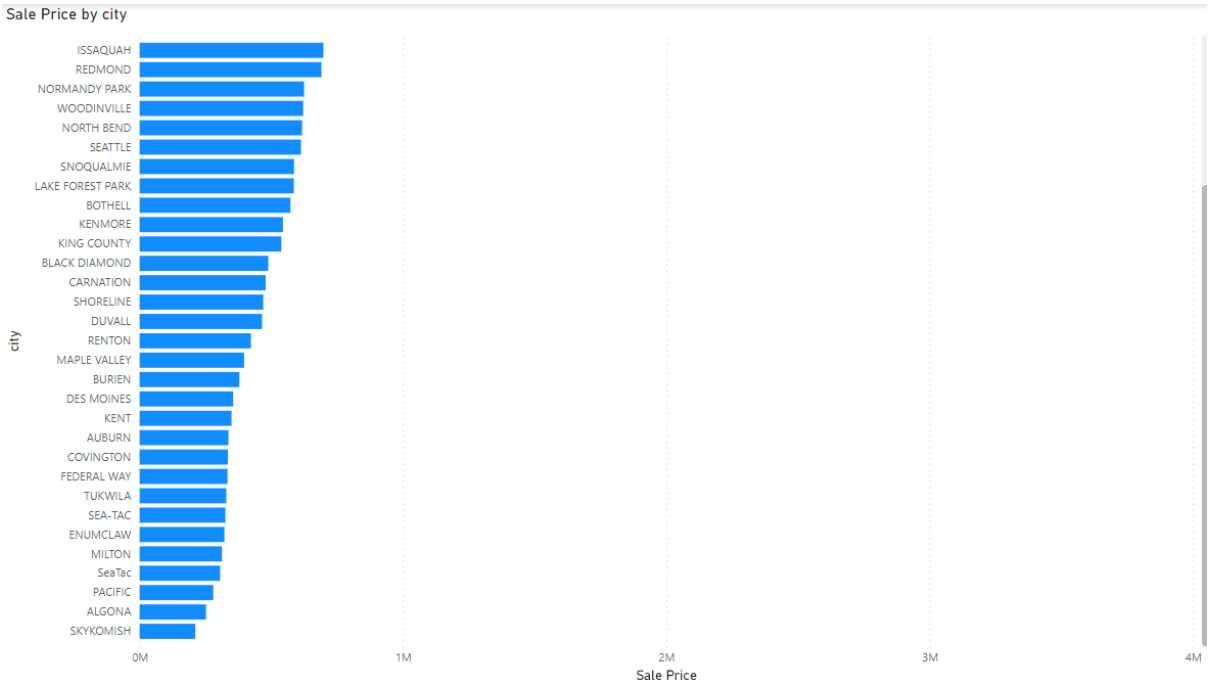


From the above graph we can observe that demand of sqft decreased a lot after 2010 and it fell drastically after 2020. This implies people are not demanding larger land rather they are interested in a good design in a smaller area. Due to this reason, HPI is increased over the years because sale price is decided by the design of the house rather than area size. Due to increase in population sellers cannot give much area to a buyer, but they are increasing the price of the house by making the house more attractive through extremely beautiful interior and exterior designing.



From the above graph we can say that submarket S is most expensive compared to other submarket and submarket L is the least expensive.





From the above graph we can say that Hunts Point is most expensive city and Skykomish is least expensive city.

MODEL CREATION

Teck Stack Used: Python

Algorithm Used: Linear Regression, Ridge Linear Regression and Lasso Linear Regression.

```
: h.isnull().sum()
```

```
: HPI                0
  sale_price         0
  city               0
  land_val           0
  sqft               0
  stories            0
  beds               0
  bath_full          0
  bath_3qtr          0
  bath_half          0
  garb_sqft          0
  gara_sqft          0
  wfnt               0
  golf               0
  greenbelt          0
  noise_traffic      0
  view_rainier       0
  view_olympics      0
  view_cascades      0
  view_territorial   0
  submarket          0
  dtype: int64
```

No null value in the data.

```
from sklearn.preprocessing import LabelEncoder
```

```
lb=LabelEncoder()
```

```
h["city"]=lb.fit_transform(h["city"])
```

```
h["submarket"]=lb.fit_transform(h["submarket"])
```

Used label encoder to give dummy variables to categorical variables.

```
h.skew()
```

```
HPI                 7.205437
sale_price          7.205437
city               -0.504100
land_val            7.960366
sqft                1.358735
stories             0.366041
beds                0.952946
bath_full           0.798974
bath_3qtr           1.182725
bath_half           0.391467
garb_sqft           3.168236
gara_sqft           0.885822
wfnt                9.237258
golf               12.571219
greenbelt           5.212632
noise_traffic       2.933678
view_rainier       12.697176
view_olympics       7.360302
view_cascades       6.777193
view_territorial    3.345806
submarket          -0.112935
dtype: float64
```

The acceptable value of skewness is considered to be between -3 and +3.

```
h["HPI"]=np.cbrt(h["HPI"])
```

```
h["Sale_price"]=np.cbrt(h["sale_price"])
```

```
h["land_val"]=np.cbrt(h["land_val"])
```

```
h["sqft"]=np.cbrt(h["sqft"])
```

```
h["stories"]=np.cbrt(h["stories"])
```

```
h["beds"]=np.cbrt(h["beds"])
```

```
h["bath_half"]=np.cbrt(h["bath_half"])
```

```
h["garb_sqft"]=np.cbrt(h["garb_sqft"])
```

```
h["gara_sqft"]=np.cbrt(h["gara_sqft"])
```

```
h["wfnt"]=np.cbrt(h["wfnt"])
```

```
h["golf"]=np.cbrt(h["golf"])
```

```
h["greenbelt"]=np.cbrt(h["greenbelt"])
```

```
h["noise_traffic"]=np.cbrt(h["noise_traffic"])
```

```
h["view_olympics"]=np.cbrt(h["view_olympics"])
```

```
h["view_cascades"]=np.cbrt(h["view_cascades"])
```

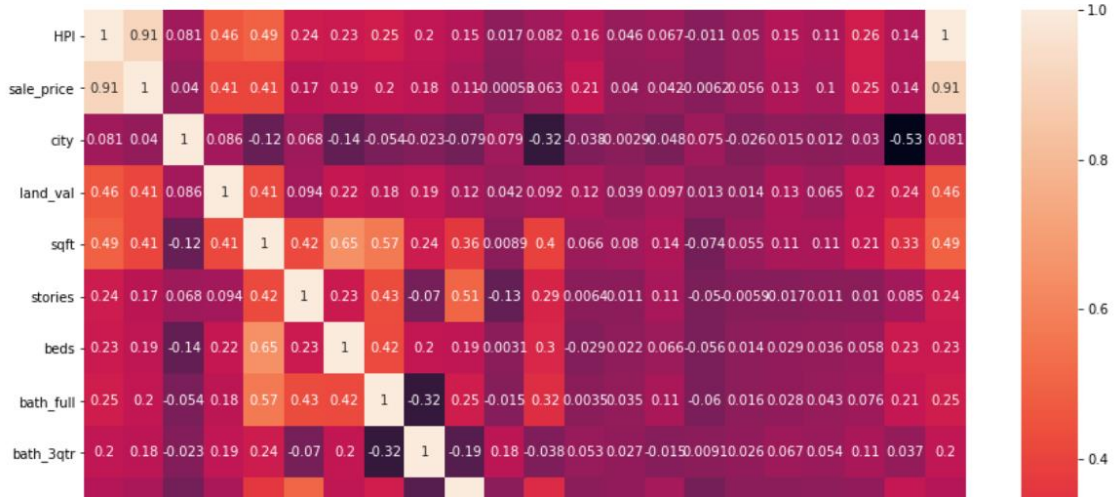
```
h["view_territorial"]=np.cbrt(h["view_territorial"])
```

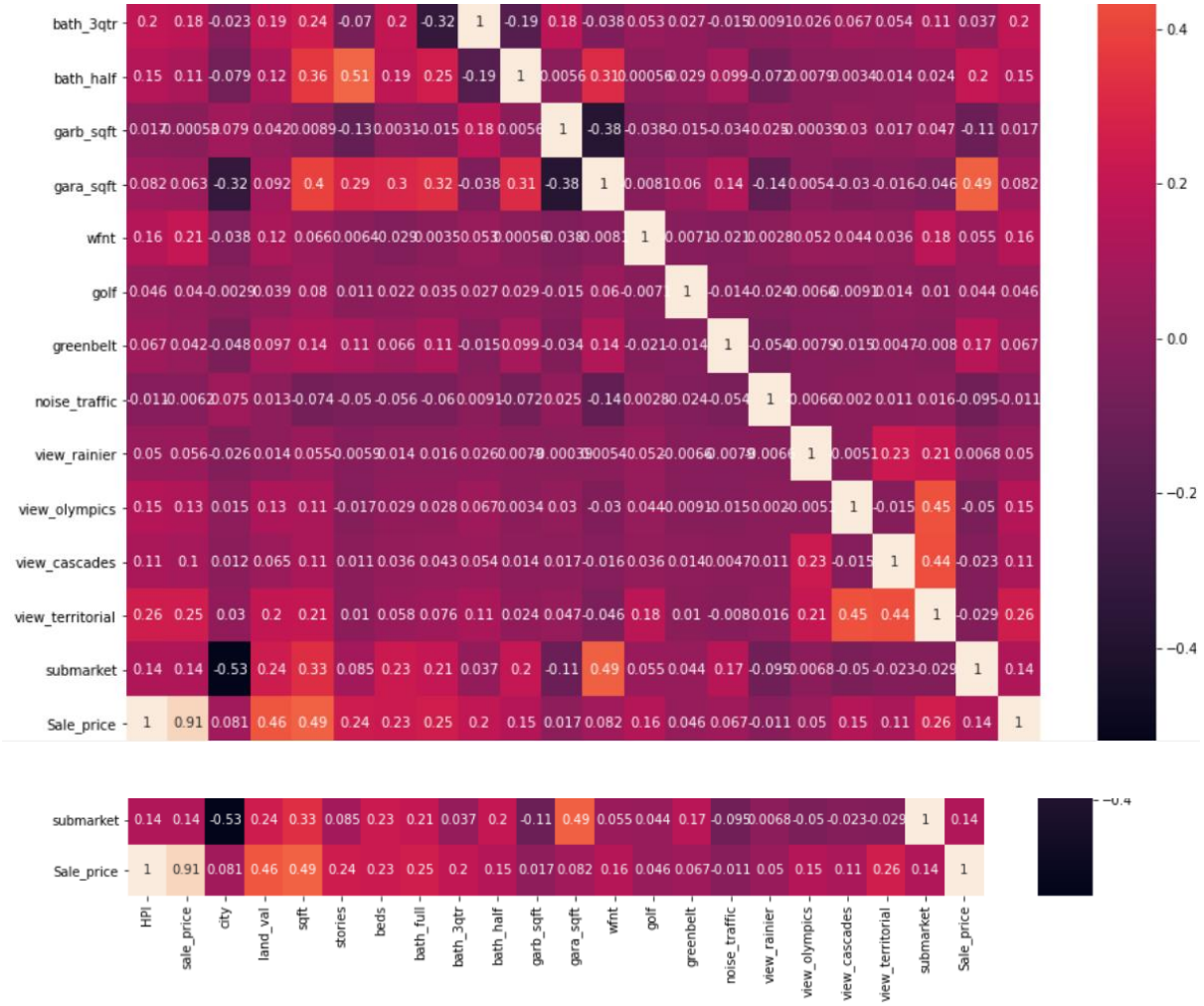
Skewness removed using CBRT method.

I have used cube root method; it is efficient in removing high skewness that is present in dataset and it can be also applied to zero and negative values. As here I am using Linear Regression hence, I should meet one of the most important assumptions of Linear Regression that is the data should be in normal distribution. I applied CBRT to all the variables to make the skewness closed to zero. So that the model works fine.

Skewness = 3(Mean-Median)/Standard Deviation

```
corr_hmap=h.corr(method="pearson")
plt.figure(figsize=(15,16))
sns.heatmap(corr_hmap,annot=True)
plt.show() # no multicollinearity found
```





No multicollinearity presents in the data, this satisfies another most important assumption of Linear Regression. In case of high multicollinearity, the model will get overfit.

```
: from scipy.stats import zscore
  z=np.abs(zscore(h))
  z
```

```
: array([[1.00105078, 0.67572058, 0.44896952, ..., 2.95291001, 0.40206748,
          1.00105078],
         [0.73184142, 0.56108364, 0.44896952, ..., 0.31262505, 1.4287006 ,
          0.73184142],
         [1.39973762, 0.81605201, 0.95968448, ..., 0.31262505, 1.50052833,
          1.39973762],
         ...,
         [1.66058199, 1.36778178, 0.95968448, ..., 0.31262505, 1.50052833,
          1.66058199],
         [3.01785545, 3.40574993, 1.85762352, ..., 3.80169131, 1.4287006 ,
          3.01785545],
         [3.01785545, 3.40574993, 1.85762352, ..., 3.80169131, 1.4287006 ,
          3.01785545]])
```

```
: thershold=3 #above 3 is considered as outliers
  print (np.where(z>3))
```

```
(array([     6,      6,     32, ..., 560217, 560217, 560217], dtype=int64), array([ 6, 12,  6, ..., 18, 19, 21], dtype=int64))
```

```
: H=h[(z<3).all(axis=1)]
```

```
: H
```

Using Z score method to remove outliers to satisfy another most important assumption of Linear Regression.

Z score formula= X- Mean/ Standard Deviation

X is the observed value.

```
x = h.drop('HPI', axis=1).copy()
```

```
y = h['HPI'].copy()
y
```

Y variable is Home Price Index or HPI

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=42)
```

```
lm=LinearRegression()
```

```
lm.fit(x_train,y_train)
```

```
LinearRegression()
```

Using Linear Regression to predict

```
x.columns
```

```
Index(['sale_price', 'city', 'land_val', 'sqft', 'stories', 'beds',
       'bath_full', 'bath_3qtr', 'bath_half', 'garb_sqft', 'gara_sqft', 'wfnt',
       'golf', 'greenbelt', 'noise_traffic', 'view_rainier', 'view_olympics',
       'view_cascades', 'view_territorial', 'submarket', 'Sale_price'],
      dtype='object')
```

These are all X variables.

```
lm.intercept_
```

```
4.537664120732796
```

```
lm.coef_
```

```
array([ 2.57733803e-06,  9.34415468e-03, -5.44089971e-08,  1.76742044e-04,
        1.60664252e-01, -9.26036654e-03,  2.16586938e-02,  6.14639215e-02,
       -3.07807745e-02,  1.01724315e-04, -3.75806755e-05, -5.06969575e-02,
        2.75011400e-02,  1.00312920e-01,  1.05770250e-03, -3.04232607e-02,
        6.15115851e-02,  2.99062736e-02,  1.68326715e-02,  7.11654615e-03])
```

```
lm.score(x_train,y_train)
```

```
0.854135283209907
```

We can see that this model is very efficient in prediction future HPI as accuracy score is 100%.

```
print("Error")
print("Mean Absolute Error:",mean_absolute_error(y_test,pred))
print("Mean Squared Error:",mean_squared_error(y_test,pred))
print("Root Mean Squared Error:",np.sqrt(mean_squared_error(y_test,pred)))
```

```
Error
Mean Absolute Error: 0.39800919148768865
Mean Squared Error: 0.3467371934365006
Root Mean Squared Error: 0.5888439465906911
```

Error is less.

```
# regularisation
from sklearn.linear_model import Lasso,Ridge
```

```
ls=Lasso(alpha=.0001)
rd=Ridge(alpha=4)
```

```
ls.fit(x_train,y_train)
print(ls.score(x_train,y_train))
predlasso=ls.predict(x_test)
print(mean_squared_error(y_test,predlasso))
```

```
0.8541341078913723
0.3467409998260632
```

Implementing Lasso regression is giving accuracy score 85%.

```
  # choosing the cross validation instead of arbitary choosing alphas=4
from sklearn.model_selection import RandomizedSearchCV
alphavalue={"alpha":[1,0.1,0.01,0.001,0.0001,0]}
model=Ridge()
rsearch=RandomizedSearchCV(estimator=model,param_distributions=alphavalue)
rsearch.fit(x_train,y_train)
print(rsearch)
print(rsearch.best_score_)
print(rsearch.best_estimator_.alpha)
print(rsearch.best_params_)
```

```
D:\ python\lib\site-packages\sklearn\model_selection\_search.py:292: UserWarning
an n_iter=10. Running 6 iterations. For exhaustive searches, use GridSearchCV.
  warnings.warn(
```

```
RandomizedSearchCV(estimator=Ridge(),
                   param_distributions={'alpha': [1, 0.1, 0.01, 0.001, 0.0001,
                                                  0]})
0.8537678743255576
1
{'alpha': 1}
```

Ridge regression is also giving 85% accuracy.

I could have used standard scaler to remove outliers but the formula of Z score and standard scaler is same. And after applying Z score the model is working fine.

Both Lasso and Ridge is regularization technique but I prefer Ridge because it helps to reduce overfitting problem by decreasing the larger coefficient whereas lasso make coefficient some variables zero that means some variables gets eliminated from the model. And Ridge regression is best when the number of variables is large.

In this case although Linear Regression is giving 85 % accuracy score yet I will choose Ridge regression as the best model because it has giving me the best model after cross validation every parameters.

This is the model I have created to predict HPI, and I would recommend Ridge Linear Regression where Apha=1 as the best model to predict HPI in future .