House Price Prediction

In [1]:
```
#import libraries

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

%matplotlib inline
```

In [2]:
```
#import dataset with pandas

house = pd.read_csv('C:\\Users\\bittu\\Desktop\\home_data.csv' , encoding="ISO
-8859-1")
```

In [3]:
```
#first 5 data from dataset
house.head()
```

Out[3]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	wa
0	7129300520	20141013T000000	221900	3	1.00	1180	5650	1.0	
1	6414100192	20141209T000000	538000	3	2.25	2570	7242	2.0	
2	5631500400	20150225T000000	180000	2	1.00	770	10000	1.0	
3	2487200875	20141209T000000	604000	4	3.00	1960	5000	1.0	
4	1954400510	20150218T000000	510000	3	2.00	1680	8080	1.0	

5 rows × 21 columns

In [4]:
```
#info about dataset
house.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21613 entries, 0 to 21612
Data columns (total 21 columns):
id               21613 non-null int64
date             21613 non-null object
price            21613 non-null int64
bedrooms         21613 non-null int64
bathrooms        21613 non-null float64
sqft_living      21613 non-null int64
sqft_lot         21613 non-null int64
floors           21613 non-null float64
waterfront       21613 non-null int64
view             21613 non-null int64
condition        21613 non-null int64
grade            21613 non-null int64
sqft_above       21613 non-null int64
sqft_basement    21613 non-null int64
yr_built         21613 non-null int64
yr_renovated     21613 non-null int64
zipcode          21613 non-null int64
lat              21613 non-null float64
long             21613 non-null float64
sqft_living15    21613 non-null int64
sqft_lot15       21613 non-null int64
dtypes: float64(4), int64(16), object(1)
memory usage: 3.5+ MB
```

In [5]:
```
# X is features of houses y is price
# we use X(capital) for two dimentional array y(small) for single dimentional
 array
X = house[['bedrooms','bathrooms','sqft_living','sqft_lot','floors','sqft_abov
e','grade','sqft_basement','yr_built','condition','zipcode']]
y = house['price']
```

In [6]:
```
X.shape #two dimentional array
```

Out[6]:
```
(21613, 11)
```

In [7]:
```
y.shape #single dimentional array
```

Out[7]:
```
(21613,)
```

In [8]:
```
# check dataset
X
```

Out[8]:

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	sqft_above	grade	sqft_basement	yr
0	3	1.00	1180	5650	1.0	1180	7	0	
1	3	2.25	2570	7242	2.0	2170	7	400	
2	2	1.00	770	10000	1.0	770	6	0	
3	4	3.00	1960	5000	1.0	1050	7	910	
4	3	2.00	1680	8080	1.0	1680	8	0	
...	...	...	...	...	...	...	...	...	
21608	3	2.50	1530	1131	3.0	1530	8	0	
21609	4	2.50	2310	5813	2.0	2310	8	0	
21610	2	0.75	1020	1350	2.0	1020	7	0	
21611	3	2.50	1600	2388	2.0	1600	8	0	
21612	2	0.75	1020	1076	2.0	1020	7	0	

21613 rows × 11 columns

In [9]:
```
# import libraries from scikit learn
from sklearn.model_selection import train_test_split
X_train , X_test , y_train , y_test = train_test_split(X,y, test_size=0.3 , ra
ndom_state=7) #random_state is optional
```

Now checking train and test dataset

In [10]:
```
X_train
```

Out[10]:

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	sqft_above	grade	sqft_basement	yr
20204	4	2.50	3148	9612	2.0	3148	9	0	
15604	3	3.50	3030	11550	2.0	3030	8	0	
2163	3	3.50	1460	1021	2.0	1150	8	310	
6338	3	2.50	2000	1950	3.0	2000	8	0	
19379	4	2.50	1950	2617	1.5	1250	7	700	
...	...	...	...	...	...	...	...	...	
919	5	1.50	2120	7700	1.5	2120	7	0	
20691	2	2.25	1060	1208	2.0	940	8	120	
5699	3	2.00	2350	5700	1.5	1810	8	540	
10742	4	2.50	2760	13093	2.0	2760	9	0	
16921	2	1.00	1100	7500	1.0	1100	7	0	

15129 rows × 11 columns

There are 15129 rows before train there is 21613 there is 70% data and 30% will be in test set

In [11]:
```
X_test
```

Out[11]:

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	sqft_above	grade	sqft_basement	yr
18732	5	1.75	1250	10530	1.0	1250	7	0	
18573	3	2.50	2660	10928	2.0	1830	9	830	
11401	3	1.75	2480	4000	1.0	1240	8	1240	
19712	3	2.50	1584	3200	2.0	1584	7	0	
17921	4	2.50	2430	3249	2.0	2430	8	0	
...	...	...	...	...	...	...	...	...	
8169	2	1.00	1050	8382	1.0	1050	7	0	
11010	3	1.00	1230	4600	1.5	1230	7	0	
739	3	2.00	1400	9177	1.0	1400	7	0	
7840	4	2.75	1930	3840	1.0	1170	7	760	
10937	4	2.25	2115	6234	2.0	2115	7	0	

6484 rows × 11 columns

We sucessfully splited data into training set and testing set

Now we are making LinerRegression Model

In [12]:
```
#import libraries from sklearn
from sklearn.linear_model import LinearRegression
model = LinearRegression()
#now we'll fit into X
model.fit(X_train,y_train)
```

Out[12]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=Fals
e)
```

In [13]:
```
#predict the prices
prediction = model.predict(X_test)
```

In [14]:
```
prediction
```

Out[14]:
```
array([197031.56526702, 812763.86299   , 753618.22994437, ...,
       297574.71157263, 310088.88856254, 329418.42878427])
```

In [15]:
```
#sucessfully predicted
```

In [16]:
```
house0 = house[house['id'] == 7129300520] # created house1 whose id is 7129300
520 in dataset
```

In [17]:
```
house0['price']
```

Out[17]:
```
0    221900
Name: price, dtype: int64
```

In [18]:
```
#now predicted price is
prediction[0]
```

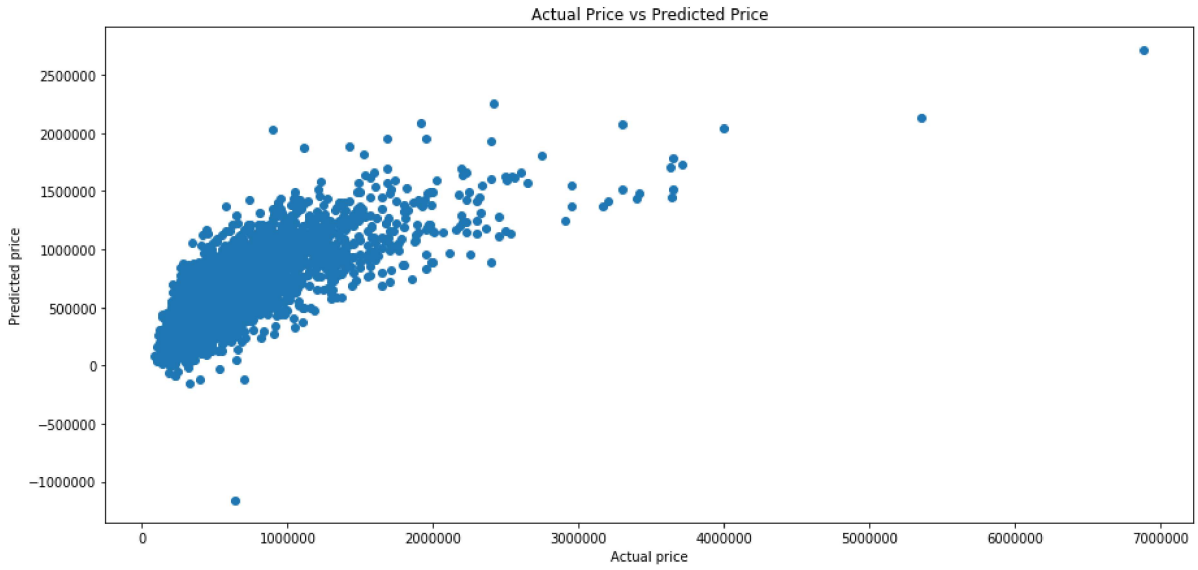
Out[18]:
```
197031.56526702363
```

Regular Price is 221900 and Predicted Price is 197031

Data visulization

In [19]:
```
plt.figure(figsize=(15,7))# just for bigger image
plt.scatter(y_test,prediction)
plt.title('Actual Price vs Predicted Price')
plt.xlabel('Actual price')
plt.ylabel('Predicted price')
```

Out[19]: Text(0, 0.5, 'Predicted price')



In [20]:
```
#Coefficient of featured (eg. bedrooms, bathrooms ..... zipcode)
model.coef_
```

Out[20]:
```
array([-5.47999393e+04,  4.96506414e+04,  1.32022773e+02, -2.47011137e-01,
        3.60909090e+04,  5.44675511e+01,  1.28169963e+05,  7.75552221e+01,
       -3.99869981e+03,  1.75394672e+04,  4.13469584e+01])
```

In [21]:
```
pd.DataFrame(model.coef_) # this is shown how our predicted price is Negative
 or Positive
```

Out[21]:

	0
0	-54799.939332
1	49650.641411
2	132.022773
3	-0.247011
4	36090.908993
5	54.467551
6	128169.962549
7	77.555222
8	-3998.699813
9	17539.467191
10	41.346958

Root Mean Square Error

In [22]:
```
#import library
from sklearn import metrics as mt
mse = mt.mean_squared_error(y_test,prediction) # mse = mean squre error
RMSE = np.sqrt(mse) # RMSE = Root Mean Squre error
```

In [23]:
```
RMSE
```

Out[23]:
```
228904.93836419654
```

In [ ]:
```

```