**House Price Prediction**

|  |
| --- |
| *#Importing the libraries* **import** pandas **as** pd **import** numpy **as** np **import** matplotlib.pyplot **as** plt **import** seaborn **as** sns **import** folium **from** folium.plugins **import** FastMarkerCluster **from** sklearn **import** preprocessing **from** sklearn **import** metrics **from** sklearn.model\_selection **import** train\_test\_split **from** sklearn.linear\_model **import** LinearRegression **from** sklearn.preprocessing **import** PolynomialFeatures **from** sklearn.preprocessing **import** StandardScaler **from** sklearn.pipeline **import** Pipeline **from** sklearn.tree **import** DecisionTreeClassifier **from** sklearn.metrics **import** mean\_squared\_error **from** sklearn.metrics **import** r2\_score **from** sklearn.metrics **import** mean\_absolute\_error **from** sklearn.linear\_model **import** Ridge |

In [ ]:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *# Importing the dataset* data **=** pd**.**read\_csv( data**.**head() |  |  |  | 'https://raw.githubusercontent.com/rashida048/Datasets/master |  |  |
| **id** | **date** | **price** | **bedrooms** | **bathrooms** | **sqft\_living** | **sqft\_lot** |

In [ ]:

Out[ ]:**f**

**0** 7129300520 20141013T000000 221900 3 1.00 1180 5650

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | 6414100192 | 20141209T000000 | 538000 | 3 | 2.25 | 2570 | 7242 |
| **2** | 5631500400 | 20150225T000000 | 180000 | 2 | 1.00 | 770 | 10000 |
| **3** | 2487200875 | 20141209T000000 | 604000 | 4 | 3.00 | 1960 | 5000 |
| **4** | 1954400510 | 20150218T000000 | 510000 | 3 | 2.00 | 1680 | 8080 |

5 rows × 21 columns

|  |
| --- |
| *#droping the unnecessary columns such as id, date, zipcode , lat and long* data**.**drop(['id','date'],axis**=**1,inplace**=True**) data**.**head() |

In [ ]:

**price bedrooms bathrooms sqft\_living sqft\_lot floors waterfront view condit**

1. 221900 3 1.00 1180 5650 1.0 0 0

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | 538000 |  | 3 | 2.25 |  | 2570 | 7242 | 2.0 | 0 | 0 |
| **2** | 180000 |  | 2 | 1.00 |  | 770 | 10000 | 1.0 | 0 | 0 |
| **3** | 604000 |  | 4 | 3.00 |  | 1960 | 5000 | 1.0 | 0 | 0 |
| **4** | 510000 |  | 3 | 2.00 |  | 1680 | 8080 | 1.0 | 0 | 0 |

|  |
| --- |
| data**.**info() |

In [ ]:

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 21613 entries, 0 to 21612 Data columns (total 19 columns):

# Column Non-Null Count Dtype --- ------ -------------- ----- 0 price 21613 non-null int64

1. bedrooms 21613 non-null int64
2. bathrooms 21613 non-null float64
3. sqft\_living 21613 non-null int64
4. sqft\_lot 21613 non-null int64
5. floors 21613 non-null float64
6. waterfront 21613 non-null int64
7. view 21613 non-null int64
8. condition 21613 non-null int64
9. grade 21613 non-null int64
10. sqft\_above 21613 non-null int64
11. sqft\_basement 21613 non-null int64
12. yr\_built 21613 non-null int64
13. yr\_renovated 21613 non-null int64
14. zipcode 21613 non-null int64
15. lat 21613 non-null float64
16. long 21613 non-null float64
17. sqft\_living15 21613 non-null int64 18 sqft\_lot15 21613 non-null int64 dtypes: float64(4), int64(15) memory usage: 3.1 MB

|  |
| --- |
| data**.**describe() |

In [ ]:

**price bedrooms bathrooms sqft\_living sqft\_lot fl**

**count** 2.161300e+04 21613.000000 21613.000000 21613.000000 2.161300e+04 21613.000

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **mean** | 5.400881e+05 | 3.370842 | 2.114757 | 2079.899736 | 1.510697e+04 |  |
| **std** | 3.671272e+05 | 0.930062 | 0.770163 | 918.440897 | 4.142051e+04 |  |
| **min** | 7.500000e+04 | 0.000000 | 0.000000 | 290.000000 | 5.200000e+02 |  |
| **25%** | 3.219500e+05 | 3.000000 | 1.750000 | 1427.000000 | 5.040000e+03 |  |
| **50%** | 4.500000e+05 | 3.000000 | 2.250000 | 1910.000000 | 7.618000e+03 |  |
| **75%** | 6.450000e+05 | 4.000000 | 2.500000 | 2550.000000 | 1.068800e+04 |  |
| **max** | 7.700000e+06 | 33.000000 | 8.000000 | 13540.000000 | 1.651359e+06 |  |

1.494

0.539

1.000

1.000

1.500

2.000

3.500

In [ ]: *# checking for null values/missing values* data**.**isnull()**.**sum()

|  |  |
| --- | --- |
| Out[ ]: | price 0 bedrooms 0 bathrooms 0 sqft\_living 0 sqft\_lot 0 floors 0 waterfront 0 view 0 condition 0 grade 0 sqft\_above 0 sqft\_basement 0 yr\_built 0 yr\_renovated 0 zipcode 0 lat 0 long 0 sqft\_living15 0 sqft\_lot15 0 dtype: int64 |

In [ ]: data**.**nunique()

price 4032 bedrooms 13 bathrooms 30 sqft\_living 1038 sqft\_lot 9782 floors 6 waterfront 2 view 5 condition 5

grade 12 sqft\_above 946 sqft\_basement 306 yr\_built 116 yr\_renovated 70 zipcode 70 lat 5034 long 752 sqft\_living15 777 sqft\_lot15 8689 dtype: int64

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | 538000 | 3 | 2 | 0.189808 | 0.004385 | 2 | 0 | 0 |
| **2** | 180000 | 2 | 1 | 0.056869 | 0.006056 | 1 | 0 | 0 |
| **3** | 604000 | 4 | 3 | 0.144756 | 0.003028 | 1 | 0 | 0 |
| **4** | 510000 | 3 | 2 | 0.124077 | 0.004893 | 1 | 0 | 0 |

# Data Preprocessing

|  |
| --- |
| *# changing float to integer*  data['bathrooms'] **=** data['bathrooms']**.**astype(int) data['floors'] **=** data['floors']**.**astype(int)  *# renaming the column yr\_built to age and changing the values to age* data**.**rename(columns**=**{'yr\_built':'age'},inplace**=True**) data['age'] **=** 2023 **-** data['age']  *# changing the column yr\_renovated to renovated and changing the values to 0 and* data**.**rename(columns**=**{'yr\_renovated':'renovated'},inplace**=True**) data['renovated'] **=** data['renovated']**.**apply(**lambda** x: 0 **if** x **==** 0 **else** 1) |

In [ ]:

In [ ]: *# using simple feature scaling* data['sqft\_living'] **=** data['sqft\_living']**/**data['sqft\_living']**.**max() data['sqft\_living15'] **=** data['sqft\_living15']**/**data['sqft\_living15']**.**max() data['sqft\_lot'] **=** data['sqft\_lot']**/**data['sqft\_lot']**.**max() data['sqft\_above'] **=** data['sqft\_above']**/**data['sqft\_above']**.**max() data['sqft\_basement'] **=** data['sqft\_basement']**/**data['sqft\_basement']**.**max() data['sqft\_lot15'] **=** data['sqft\_lot15']**/**data['sqft\_lot15']**.**max()

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| data**.**head() |  |  |  |  |  |  |
| **price bedrooms** | **bathrooms** | **sqft\_living** | **sqft\_lot** | **floors** | **waterfront** | **view cond** |

In [ ]: Out[ ]:

**0** 221900 3 1 0.087149 0.003421 1 0 0

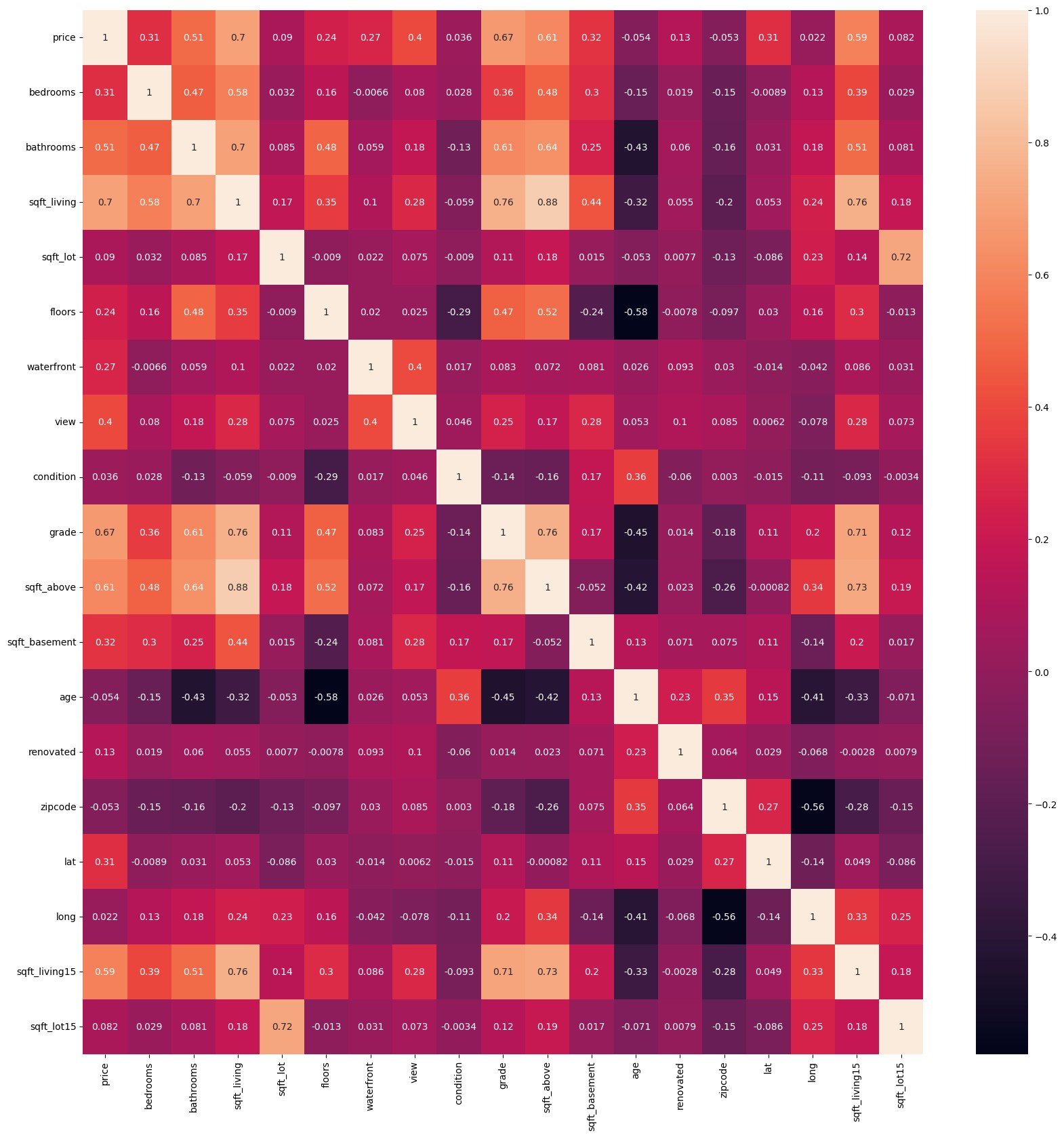
# Exploratory Data Analysis

**Correlation Matrix to find the relationship between the variables**

In [ ]: *# using correlation statistical method to find the relation between the price an* data**.**corr()['price']**.**sort\_values(ascending**=False**)

Out[ ]: price 1.000000 sqft\_living 0.702035 grade 0.667434 sqft\_above 0.605567 sqft\_living15 0.585379 bathrooms 0.510072 view 0.397293 sqft\_basement 0.323816 bedrooms 0.308350 lat 0.307003 waterfront 0.266369 floors 0.237211 renovated 0.126092 sqft\_lot 0.089661 sqft\_lot15 0.082447 condition 0.036362 long 0.021626 zipcode -0.053203 age -0.054012 Name: price, dtype: float64

In [ ]: plt**.**figure(figsize**=**(20,20)) sns**.**heatmap(data**.**corr(),annot**=True**) plt**.**show()

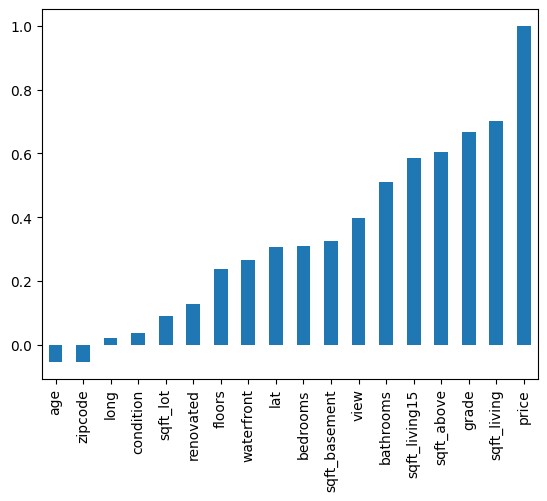


## Visualizing the coorelation with price

|  |
| --- |
| data**.**corr()['price'][:**-**1]**.**sort\_values()**.**plot(kind**=**'bar') |

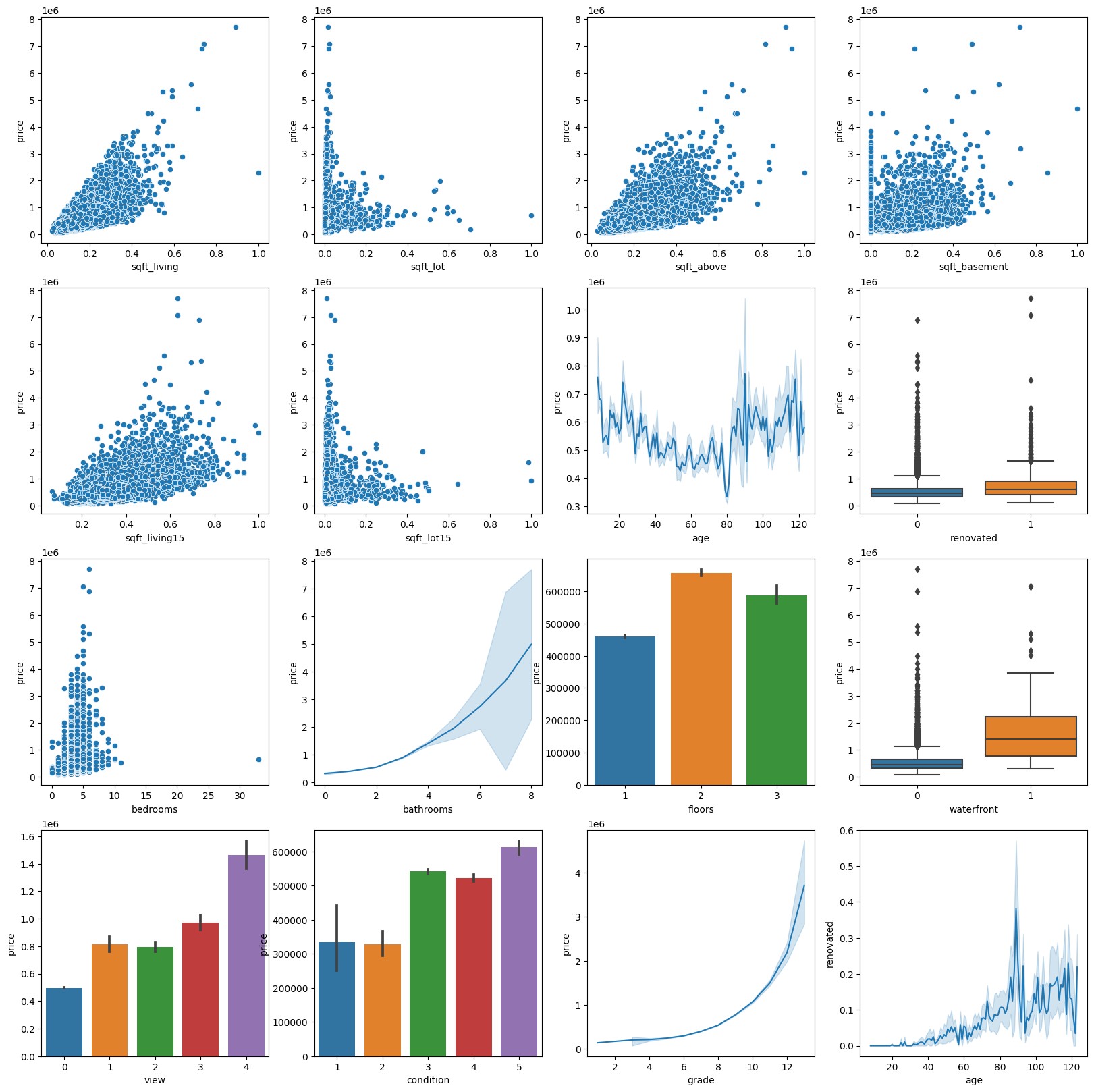
In [ ]:

Out[ ]: <Axes: >



|  |
| --- |
| *# visualizing the relation between price and sqft\_living, sqft\_lot, sqft\_above,* fig, ax **=** plt**.**subplots(4,4,figsize**=**(20,20)) sns**.**scatterplot( x **=** data['sqft\_living'], y **=** data['price'],ax**=**ax[0,0]) sns**.**scatterplot( x **=** data['sqft\_lot'], y **=** data['price'],ax**=**ax[0,1]) sns**.**scatterplot( x **=** data['sqft\_above'], y **=** data['price'],ax**=**ax[0,2]) sns**.**scatterplot( x **=** data['sqft\_basement'], y **=** data['price'],ax**=**ax[0,3]) sns**.**scatterplot( x **=** data['sqft\_living15'], y **=** data['price'],ax**=**ax[1,0]) sns**.**scatterplot( x **=** data['sqft\_lot15'], y **=** data['price'],ax**=**ax[1,1]) sns**.**lineplot( x **=** data['age'], y **=** data['price'],ax**=**ax[1,2]) sns**.**boxplot( x **=** data['renovated'], y **=** data['price'],ax**=**ax[1,3]) sns**.**scatterplot( x **=** data['bedrooms'], y **=** data['price'],ax**=**ax[2,0]) sns**.**lineplot( x **=** data['bathrooms'], y **=** data['price'],ax**=**ax[2,1]) sns**.**barplot( x **=** data['floors'], y **=** data['price'],ax**=**ax[2,2]) sns**.**boxplot( x **=** data['waterfront'], y **=** data['price'],ax**=**ax[2,3]) sns**.**barplot( x **=** data['view'], y **=** data['price'],ax**=**ax[3,0]) sns**.**barplot( x **=** data['condition'], y **=** data['price'],ax**=**ax[3,1]) sns**.**lineplot( x **=** data['grade'], y **=** data['price'],ax**=**ax[3,2]) sns**.**lineplot( x **=** data['age'], y **=** data['renovated'],ax**=**ax[3,3]) plt**.**show() |

**Visulaizing the data** In [ ]:



**Plotting the location of the houses based on longitude and latitude on the map**

In [ ]:

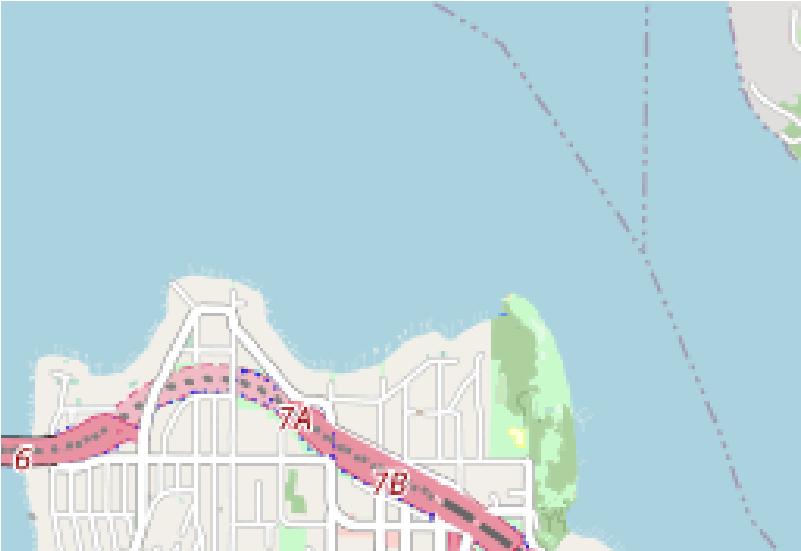
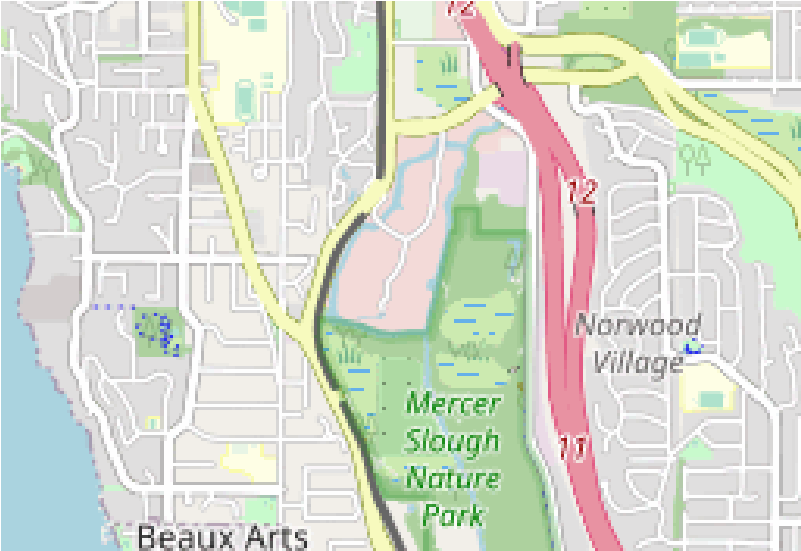
|  |
| --- |
| *# adding a new column price\_range and categorizing the price into 4 categories* data['price\_range'] **=** pd**.**cut(data['price'],bins**=**[0,321950,450000,645000,1295648 |

|  |
| --- |
| map **=** folium**.**Map(location**=**[47.5480, **-**121.9836],zoom\_start**=**8) marker\_cluster **=** FastMarkerCluster(data[['lat', 'long']]**.**values**.**tolist())**.**add\_to map |

] In [ ]:

Out[ ]:

Make this Notebook Trusted to load map: File -> Trust Notebook



22

13

34

47

6

3

25

36

35

56

7

30

52

+

−

[Leafl](http://www.openstreetmap.org/copyright)

[e](http://www.openstreetmap.org/copyright)

[t](http://www.openstreetmap.org/copyright)

[(https://leafletjs.co](http://www.openstreetmap.org/copyright)

[m](http://www.openstreetmap.org/copyright)

[)](http://www.openstreetmap.org/copyright)

[|](http://www.openstreetmap.org/copyright)

[Data by ©](http://www.openstreetmap.org/copyright)

[OpenStreetM](http://www.openstreetmap.org/copyright)

[a](http://www.openstreetmap.org/copyright)

[p](http://www.openstreetmap.org/copyright)

[(http://openstreetmap.or](http://www.openstreetmap.org/copyright)

[)](http://www.openstreetmap.org/copyright)

[g](http://www.openstreetmap.org/copyright)

[, under](http://www.openstreetmap.org/copyright)

[ODb](http://www.openstreetmap.org/copyright)

[L](http://www.openstreetmap.org/copyright)

<http://www.openstreetmap.org/copyright>

[)](http://www.openstreetmap.org/copyright)

[(](http://www.openstreetmap.org/copyright)

[.](http://www.openstreetmap.org/copyright)

# Train/Test Split

|  |
| --- |
| data**.**drop(['price\_range'],axis**=**1,inplace**=True**)  X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(data**.**drop('price',axis**=**1),da |

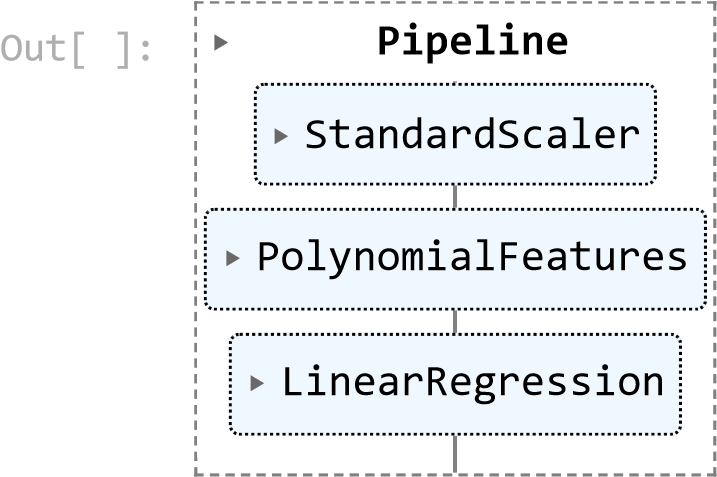
In [ ]:

# Model Training

## Using pipeline to combine the transformers and estimators and fit the model

|  |
| --- |
| input **=** [('scale',StandardScaler()),('polynomial', PolynomialFeatures(degree**=**2)) pipe **=** Pipeline(input) pipe |

In [ ]:



|  |
| --- |
| *#training the model* pipe**.**fit(X\_train,y\_train) pipe**.**score(X\_test,y\_test) |

In [ ]:

Out[ ]: 0.8271896429378042

|  |
| --- |
| *#testing the model*  pipe\_pred **=** pipe**.**predict(X\_test) r2\_score(y\_test,pipe\_pred) |

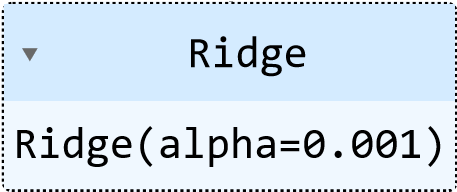
In [ ]:

Out[ ]: 0.8271896429378042

## Ridge Regression

|  |
| --- |
| Ridgemodel **=** Ridge(alpha **=** 0.001)  Ridgemodel |

In [ ]:

Out[ ]: 

|  |
| --- |
| *# training the model*  Ridgemodel**.**fit(X\_train,y\_train) Ridgemodel**.**score(X\_test,y\_test) |

In [ ]:

|  |
| --- |
| *#testing the model*  r\_pred **=** Ridgemodel**.**predict(X\_test) r2\_score(y\_test,r\_pred) |

In [ ]:

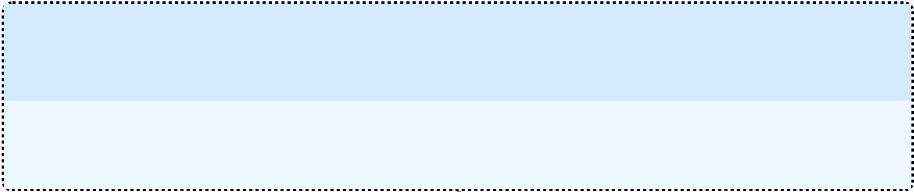
Out[ ]: 0.7123220593275169

## Random Forest Regression

|  |
| --- |
| **from** sklearn.ensemble **import** RandomForestRegressor regressor **=** RandomForestRegressor(n\_estimators**=**100, random\_state**=**0) regressor |

In [ ]:

Out[ ]: ▾ RandomForestRegressor

RandomForestRegressor(random\_state=0)

|  |
| --- |
| *# training the model* regressor**.**fit(X\_train,y\_train) regressor**.**score(X\_test,y\_test) |

In [ ]:

|  |
| --- |
| *#testing the model*  yhat **=** regressor**.**predict(X\_test) r2\_score(y\_test,yhat) |

Out[ ]: 0.878968081057204 In [ ]:

Out[ ]: 0.878968081057204

# Model Evalution

## Distribution plot from the models predictions and the actual values

|  |
| --- |
| *# displot of the actual price and predicted price for all models* fig, ax **=** plt**.**subplots(1,3,figsize**=**(20,5)) sns**.**distplot(y\_test,ax**=**ax[0]) sns**.**distplot(pipe\_pred,ax**=**ax[0]) |

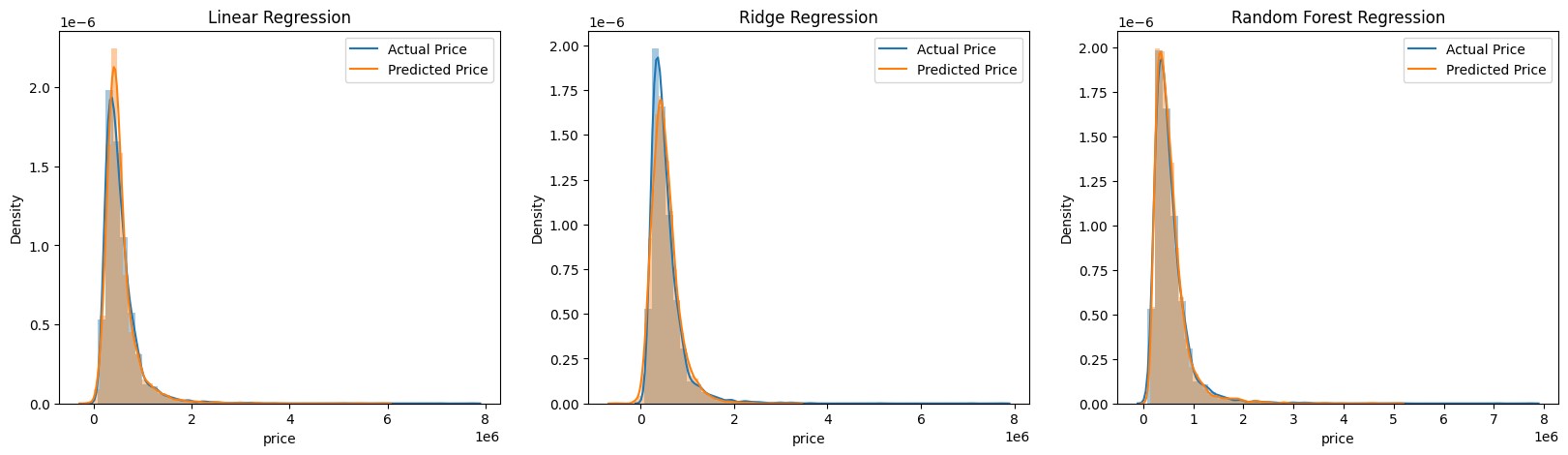
In [ ]:

sns**.**distplot(y\_test,ax**=**ax[1]) sns**.**distplot(r\_pred,ax**=**ax[1]) sns**.**distplot(y\_test,ax**=**ax[2]) sns**.**distplot(yhat,ax**=**ax[2])

*# legends*

ax[0]**.**legend(['Actual Price','Predicted Price']) ax[1]**.**legend(['Actual Price','Predicted Price']) ax[2]**.**legend(['Actual Price','Predicted Price'])

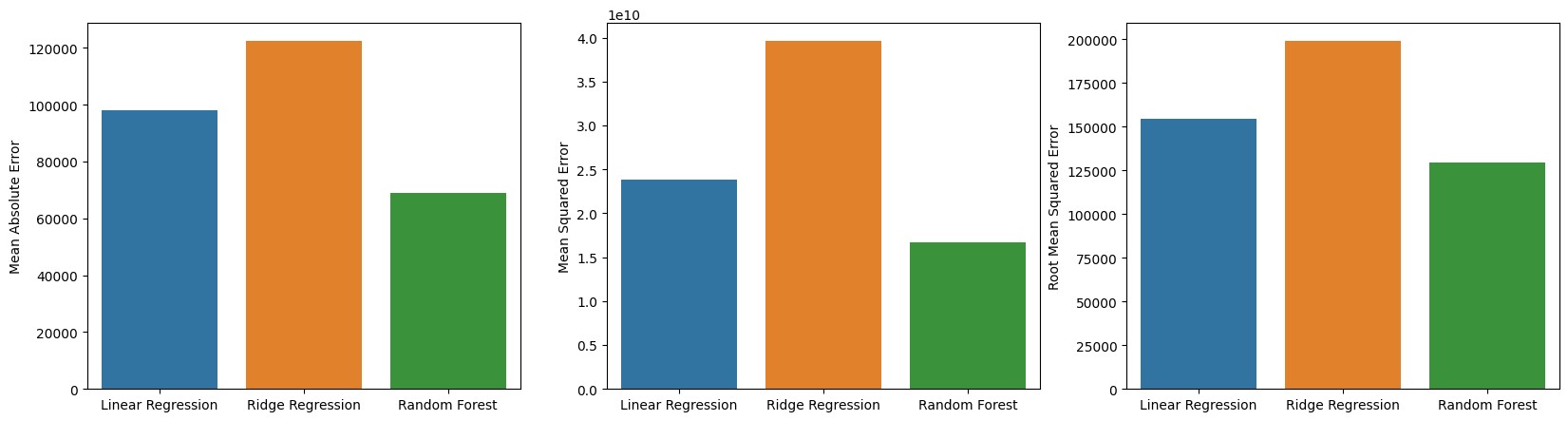
*#model name as title* ax[0]**.**set\_title('Linear Regression') ax[1]**.**set\_title('Ridge Regression') ax[2]**.**set\_title('Random Forest Regression') plt**.**show()



## Error Evaluation

|  |
| --- |
| *#plot the graph to compare mae, mse, rmse for all models* fig, ax **=** plt**.**subplots(1,3,figsize**=**(20,5)) sns**.**barplot(x**=**['Linear Regression','Ridge Regression','Random Forest'],y**=**[mean\_a sns**.**barplot(x**=**['Linear Regression','Ridge Regression','Random Forest'],y**=**[mean\_s sns**.**barplot(x**=**['Linear Regression','Ridge Regression','Random Forest'],y**=**[np**.**sqr  *# label for the graph*  ax[0]**.**set\_ylabel('Mean Absolute Error') ax[1]**.**set\_ylabel('Mean Squared Error') ax[2]**.**set\_ylabel('Root Mean Squared Error') plt**.**show() |

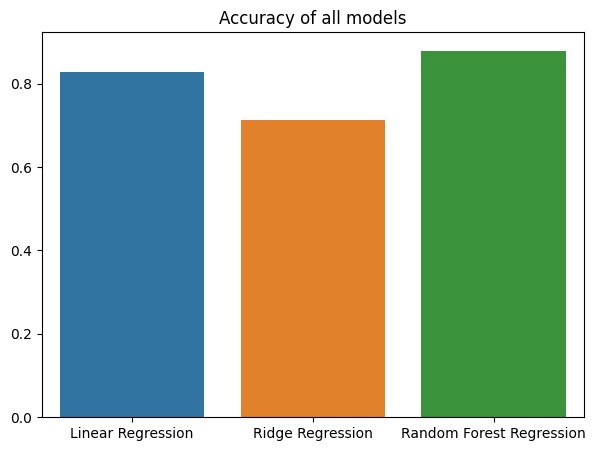
In [ ]:



**Accuracy Evaluation**

|  |
| --- |
| *# plot accuracy of all models in the same graph* fig, ax **=** plt**.**subplots(figsize**=**(7,5)) sns**.**barplot(x**=**['Linear Regression','Ridge Regression','Random Forest Regression' ax**.**set\_title('Accuracy of all models') plt**.**show() |

In [ ]:



**Predicting the price of a new house**

|  |
| --- |
| *#input the values* bedrooms **=** 3 bathrooms **=** 2 sqft\_living **=** 2000 sqft\_lot **=** 10000 floors **=** 2 waterfront **=** 0 view **=** 0 condition **=** 3 grade **=** 8 sqft\_above **=** 2000 sqft\_basement **=** 0 yr\_built **=** 1990 yr\_renovated **=** 0 zipcode **=** 98001 lat **=** 47.5480 long **=** **-**121.9836 sqft\_living15 **=** 2000 sqft\_lot15 **=** 10000 |

In [ ]:

|  |
| --- |
| *#predicting the price using random forest regression*  price **=** regressor**.**predict([[bedrooms,bathrooms,sqft\_living,sqft\_lot,floors,water print('The price of the house is $',price[0]) |

In [ ]:

The price of the house is $ 1078694.0533333335

# Conclusion

From the analysis, we can see that the Random Forest Regression model performed better than the Ridge Regression model and Polynomial Regression model.

During the EDA process, we found out that the location of the house is a very important factor in determining the price of the house, since houese with similar area and other features can have different prices depending on the location of the house.

The location of the houses has been plotted on the map using the longitude and latitude values which makesrole of location in determining the price of the house more clear.

**NOTE: For some reasons, the map was not rendered properly when the notebook was converted into pdf. So here is the image of the rendered map showing the locations of the houses, color coded according to their price range**

