# House Price Prediction

In [ ]:
```
#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('https://raw.githubusercontent.com/rashida048/Datasets/master
data.head()
```

Out[ ]:

| | id | date | price | bedrooms | bathrooms | sqft_living | sqft_lot |
|---|---|---|---|---|---|---|---|
| **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

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

Out[ ]:

| | price | bedrooms | bathrooms | sqft_living | sqft_lot | floors | waterfront | view | condi |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 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 | |

In [ ]:
```
data.info()
```

```
<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
```

In [ ]:
```
data.describe()
```

Out[ ]:

| | price | bedrooms | bathrooms | sqft_living | sqft_lot | fl |
|---|---|---|---|---|---|---|
| **count** | 2.161300e+04 | 21613.000000 | 21613.000000 | 21613.000000 | 2.161300e+04 | 21613.00 |
| **mean** | 5.400881e+05 | 3.370842 | 2.114757 | 2079.899736 | 1.510697e+04 | 1.49 |
| **std** | 3.671272e+05 | 0.930062 | 0.770163 | 918.440897 | 4.142051e+04 | 0.53 |
| **min** | 7.500000e+04 | 0.000000 | 0.000000 | 290.000000 | 5.200000e+02 | 1.00 |
| **25%** | 3.219500e+05 | 3.000000 | 1.750000 | 1427.000000 | 5.040000e+03 | 1.00 |
| **50%** | 4.500000e+05 | 3.000000 | 2.250000 | 1910.000000 | 7.618000e+03 | 1.50 |
| **75%** | 6.450000e+05 | 4.000000 | 2.500000 | 2550.000000 | 1.068800e+04 | 2.00 |
| **max** | 7.700000e+06 | 33.000000 | 8.000000 | 13540.000000 | 1.651359e+06 | 3.50 |

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()
```

Out[ ]:
```
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
```

# Data Preprocessing

In [ ]:
```python
# 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 [ ]:
```python
# 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()
```

In [ ]:
```python
data.head()
```

Out[ ]:

| | price | bedrooms | bathrooms | sqft_living | sqft_lot | floors | waterfront | view | cond |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 221900 | 3 | 1 | 0.087149 | 0.003421 | 1 | 0 | 0 | |
| **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 | |

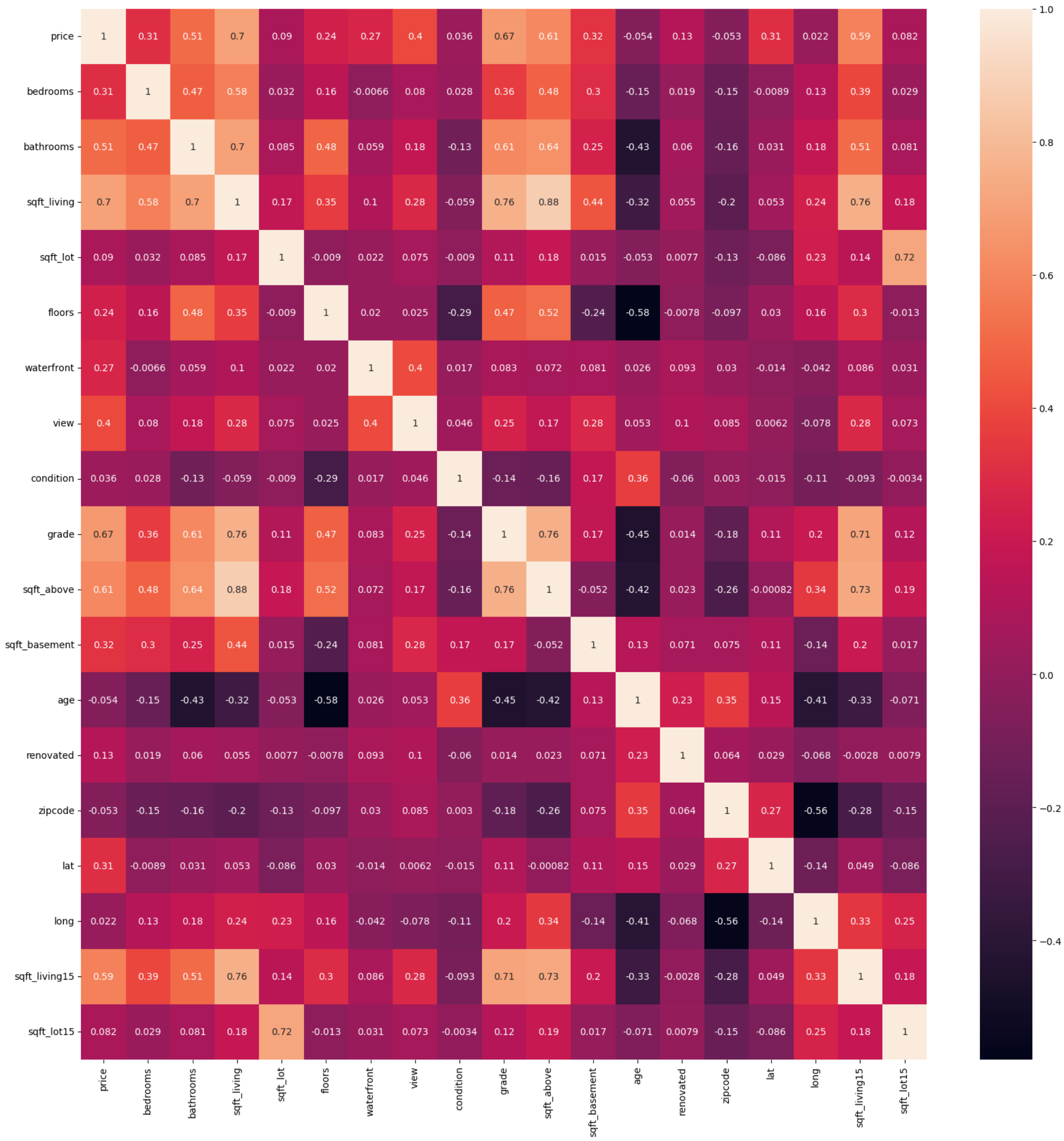
# Exploratory Data Analysis

### 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
```

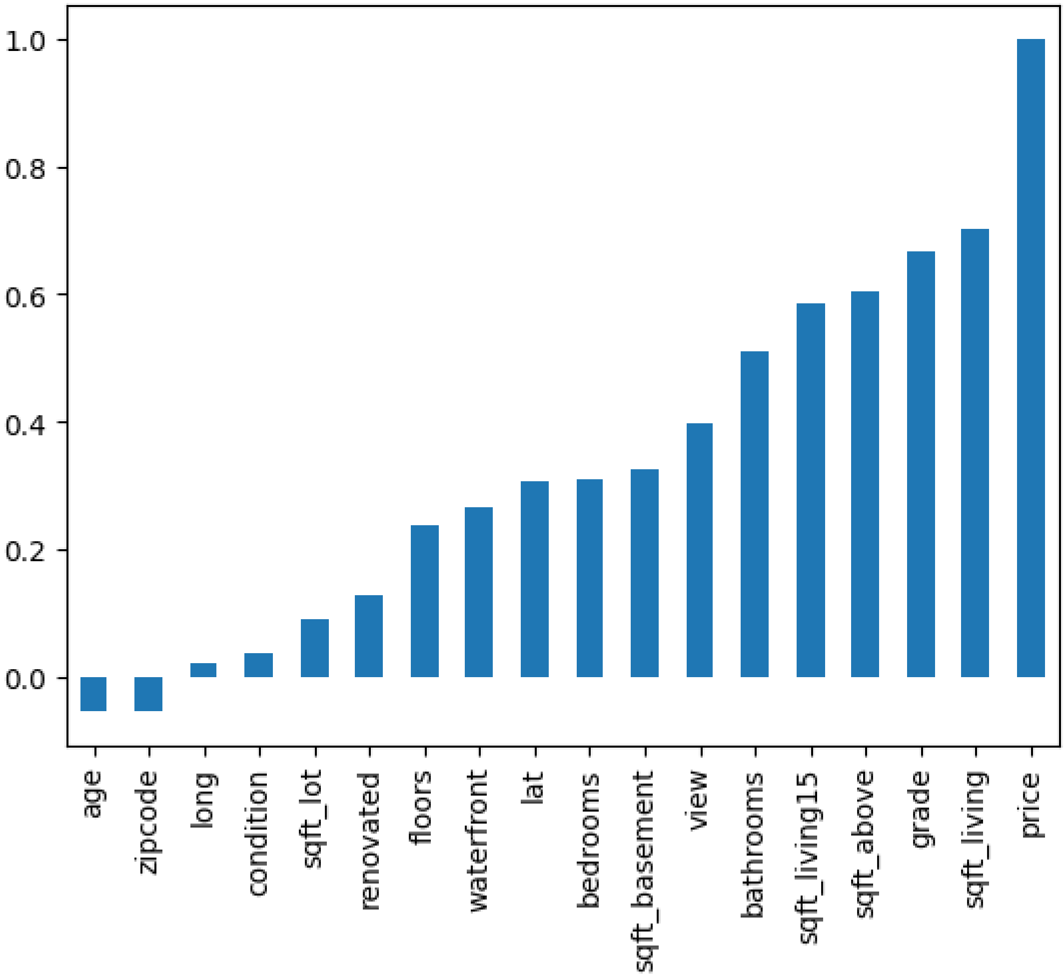
In [ ]:
```
plt.figure(figsize=(20,20))
sns.heatmap(data.corr(),annot=True)
plt.show()
```




## Visualizing the coorelation with price

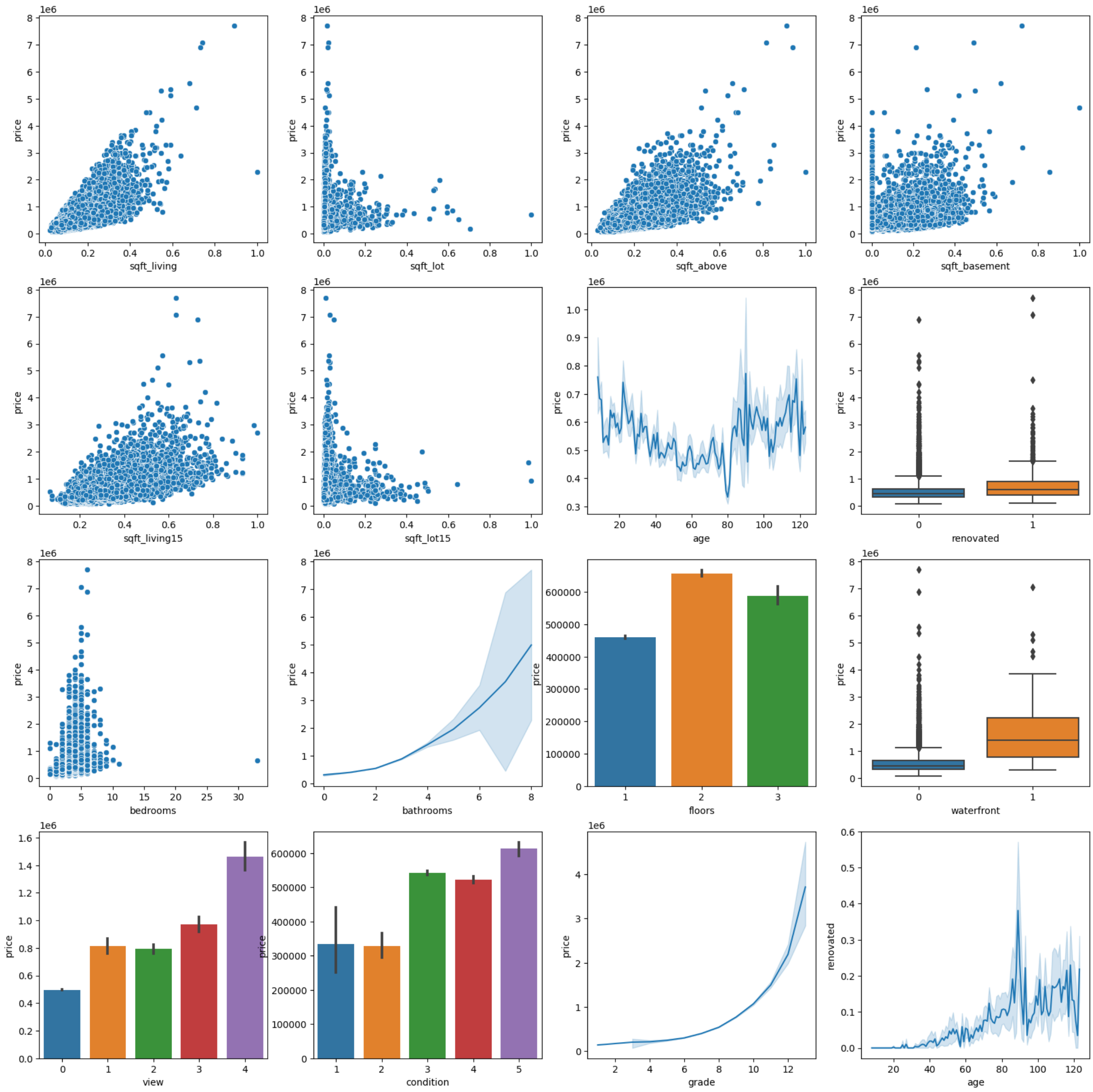
In [ ]:
```
data.corr()['price'][:-1].sort_values().plot(kind='bar')
```

Out[ ]: <Axes: >




## Visulaizing the data

In [ ]:
```
# 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()
```

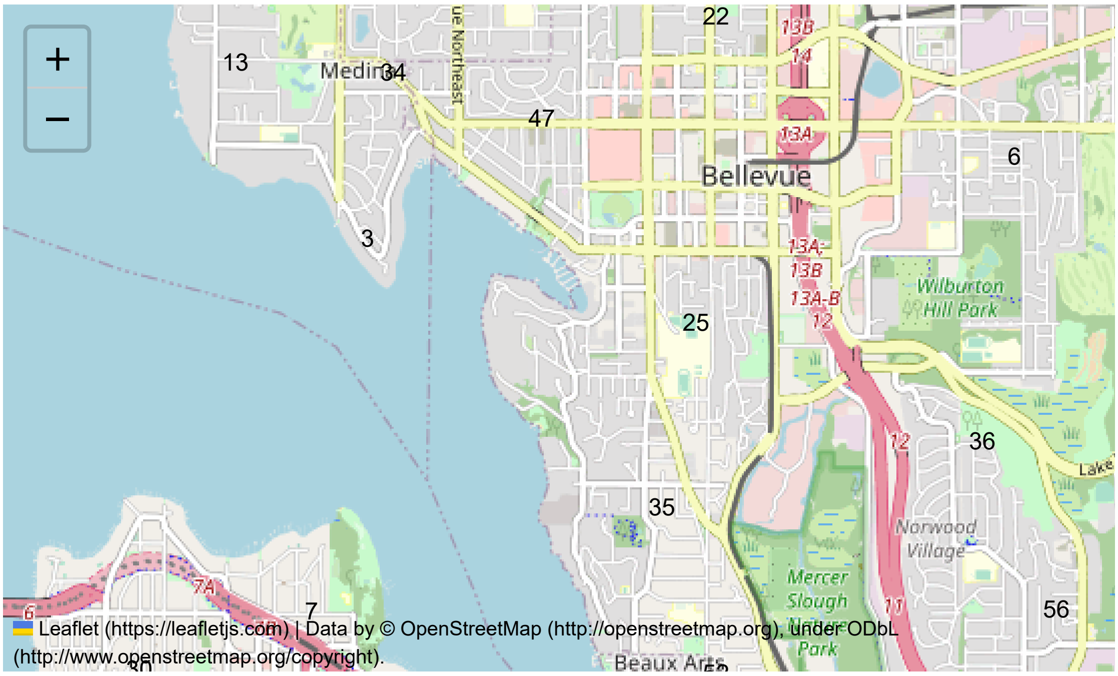



## Plotting the location of the houses based on longitude and latitude on the map

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]
```

In [ ]:
```
map = folium.Map(location=[47.5480, -121.9836],zoom_start=8)
marker_cluster = FastMarkerCluster(data[['lat', 'long']].values.tolist()).add_tc
map
```

Out[ ]:




## Train/Test Split

In [ ]:

```
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
```

## Model Training

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

In [ ]:

```
input = [('scale',StandardScaler()),('polynomial', PolynomialFeatures(degree=2))
pipe = Pipeline(input)
pipe
```

Out[ ]:

```
Pipeline
  StandardScaler
  PolynomialFeatures
  LinearRegression
```

In [ ]:

```
#training the model
pipe.fit(X_train,y_train)
pipe.score(X_test,y_test)
```

Out[ ]: 0.8271896429378042

In [ ]:

```
#testing the model
pipe_pred = pipe.predict(X_test)
r2_score(y_test,pipe_pred)
```

Out[ ]: 0.8271896429378042

## Ridge Regression

In [ ]:
```
Ridgemodel = Ridge(alpha = 0.001)
Ridgemodel
```

Out[ ]:
```
▾ Ridge
Ridge(alpha=0.001)
```

In [ ]:
```
# 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)
```

Out[ ]: 0.7123220593275169

## Random Forest Regression

In [ ]:
```
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators=100, random_state=0)
regressor
```

Out[ ]:
```
▾ RandomForestRegressor
RandomForestRegressor(random_state=0)
```

In [ ]:
```
# training the model
regressor.fit(X_train,y_train)
regressor.score(X_test,y_test)
```

Out[ ]: 0.878968081057204

In [ ]:
```
#testing the model
yhat = regressor.predict(X_test)
r2_score(y_test,yhat)
```

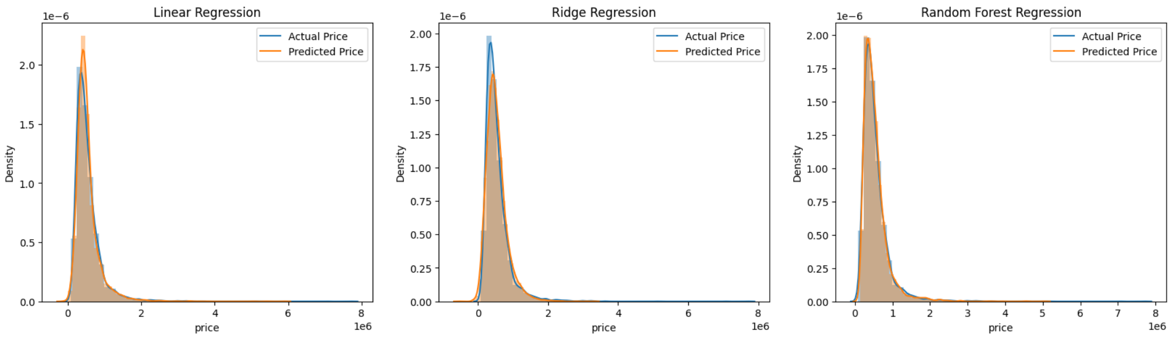
Out[ ]: 0.878968081057204

# Model Evalution

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

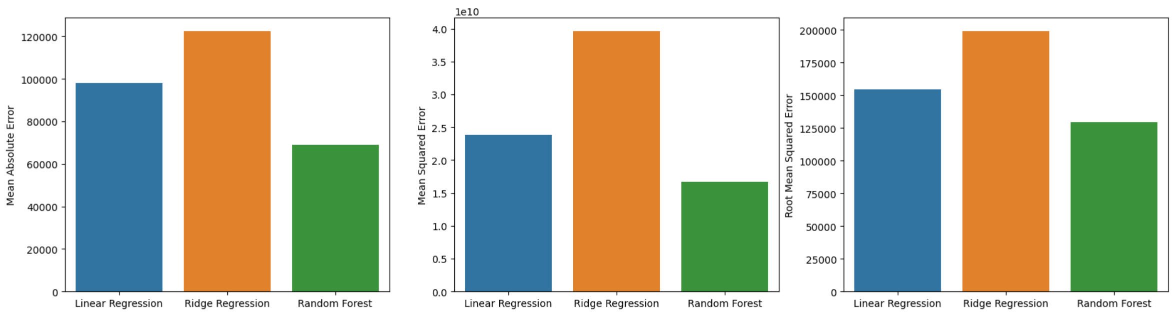
In [ ]:
```
# displot of the actual price and predicted price for all models
fig, ax = plt.subplots(1,3,figsize=(20,5))
sns.distplot(y_test,ax=ax[0])
sns.distplot(pipe_pred,ax=ax[0])
```

```
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()
```



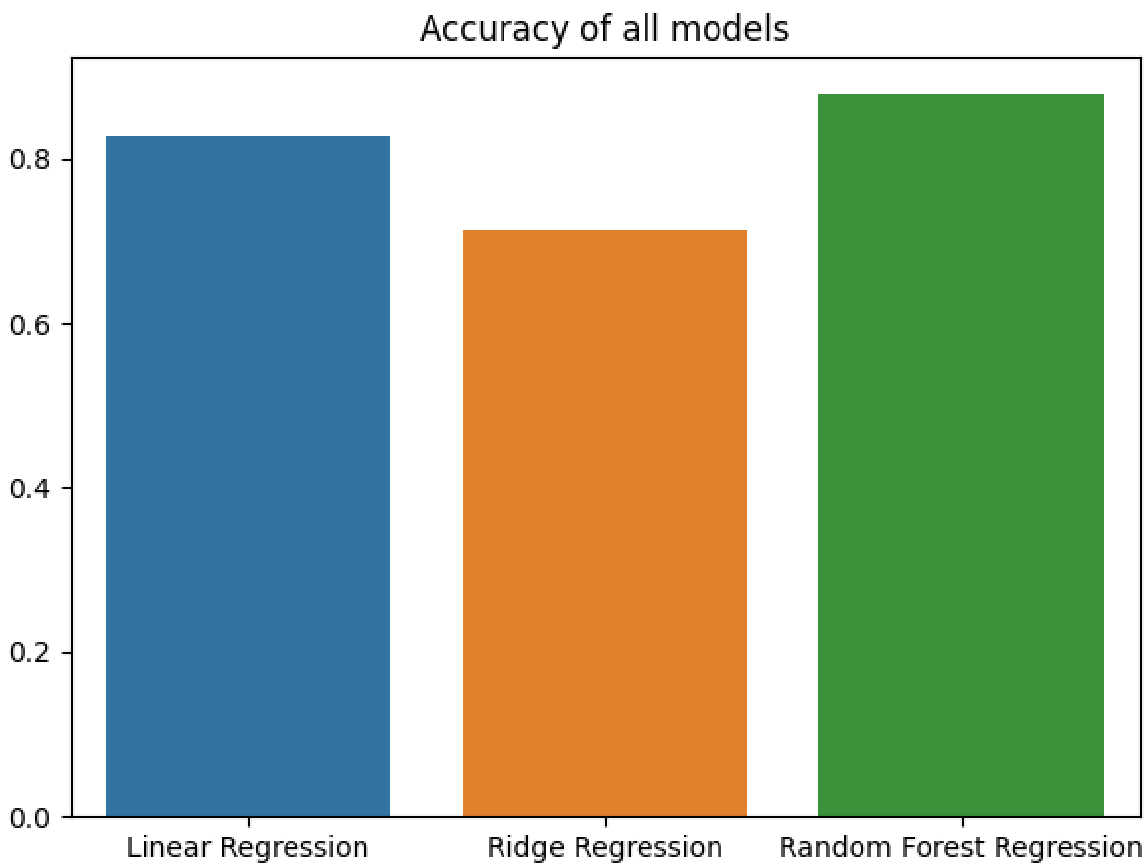

## Error Evaluation

In [ ]:
```
#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()
```




## Accuracy Evaluation

In [ ]:
```
# 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()
```




## Predicting the price of a new house

In [ ]:
```
#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])
```

```
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**