In [25]:

```python
#import all libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [26]:

```python
df = pd.read_csv("data.csv")
```

In [27]:

```python
df.head()
```

Out[27]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition
0	2014-05-02 00:00:00	313000.0	3.0	1.50	1340	7912	1.5	0	0	3
1	2014-05-02 00:00:00	2384000.0	5.0	2.50	3650	9050	2.0	0	4	5
2	2014-05-02 00:00:00	342000.0	3.0	2.00	1930	11947	1.0	0	0	4
3	2014-05-02 00:00:00	420000.0	3.0	2.25	2000	8030	1.0	0	0	4
4	2014-05-02 00:00:00	550000.0	4.0	2.50	1940	10500	1.0	0	0	4

In [ ]:

In [28]:

```python
df.describe()
```

Out[28]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront
count	4.600000e+03	4600.000000	4600.000000	4600.000000	4.600000e+03	4600.000000	4600.000000
mean	5.519630e+05	3.400870	2.160815	2139.346957	1.485252e+04	1.512065	0.007174
std	5.638347e+05	0.908848	0.783781	963.206916	3.588444e+04	0.538288	0.084404
min	0.000000e+00	0.000000	0.000000	370.000000	6.380000e+02	1.000000	0.000000
25%	3.228750e+05	3.000000	1.750000	1460.000000	5.000750e+03	1.000000	0.000000
50%	4.609435e+05	3.000000	2.250000	1980.000000	7.683000e+03	1.500000	0.000000
75%	6.549625e+05	4.000000	2.500000	2620.000000	1.100125e+04	2.000000	0.000000
max	2.659000e+07	9.000000	8.000000	13540.000000	1.074218e+06	3.500000	1.000000

In [29]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4600 entries, 0 to 4599
Data columns (total 18 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   date           4600 non-null   object 
 1   price          4600 non-null   float64
 2   bedrooms       4600 non-null   float64
 3   bathrooms      4600 non-null   float64
 4   sqft_living    4600 non-null   int64  
 5   sqft_lot       4600 non-null   int64  
 6   floors         4600 non-null   float64
 7   waterfront     4600 non-null   int64  
 8   view           4600 non-null   int64  
 9   condition      4600 non-null   int64  
 10  sqft_above     4600 non-null   int64  
 11  sqft_basement  4600 non-null   int64  
 12  yr_built       4600 non-null   int64  
 13  yr_renovated   4600 non-null   int64  
 14  street         4600 non-null   object 
 15  city           4600 non-null   object 
 16  statezip       4600 non-null   object 
 17  country        4600 non-null   object 
dtypes: float64(4), int64(9), object(5)
memory usage: 647.0+ KB
```

In [30]:

```
df.isna().sum()
```

Out[30]:

```
date             0
price            0
bedrooms         0
bathrooms        0
sqft_living      0
sqft_lot         0
floors           0
waterfront       0
view             0
condition        0
sqft_above       0
sqft_basement    0
yr_built         0
yr_renovated     0
street           0
city             0
statezip         0
country          0
dtype: int64
```

In [ ]:

```

```

In [65]:

```
df['price'] = df['price'].astype("int64")
df['bedrooms'] = df['bedrooms'].astype('int64')
```

```
df['bathrooms'] = df['bathrooms'].astype('int64')
df['floors'] = df['floors'].astype('int64')
```

In [66]:

```
df.isna().sum()
```

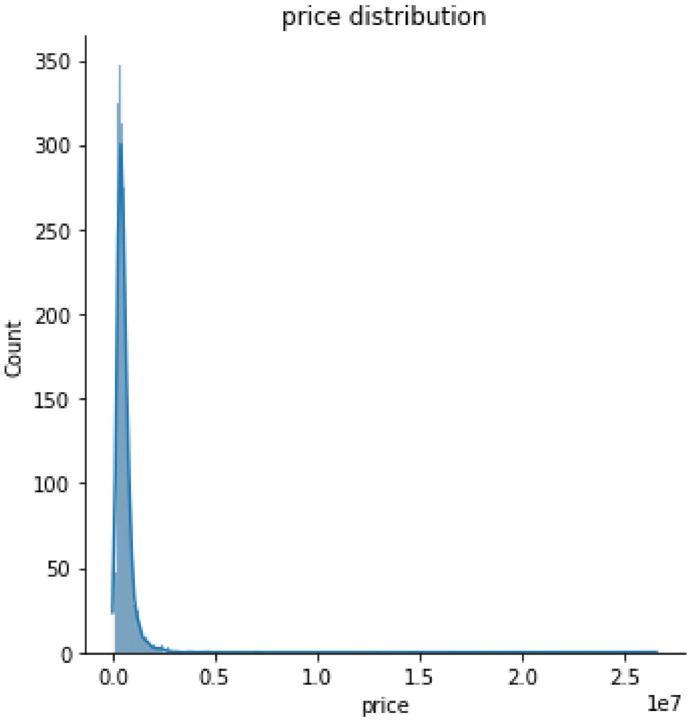
Out[66]:

```
date             0
price            0
bedrooms         0
bathrooms        0
sqft_living      0
sqft_lot         0
floors           0
waterfront       0
view             0
condition        0
sqft_above       0
sqft_basement    0
yr_built         0
yr_renovated     0
street           0
city             0
statezip         0
country          0
dtype: int64
```

In [67]:

```
plt.figure(figsize=(15, 5))
sns.displot(df['price'], kde=True)
plt.title('price distribution')
plt.rcParams['figure.figsize'] = 20,10
```

```
<Figure size 1080x360 with 0 Axes>
```



In [68]:

```
X = df.iloc[:, [1,2,6]].values
y = df.iloc[:, -17].values
```

In [69]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0, test_size=0.3
```

In [70]:

```
#from sklearn.preprocessing import StandardScaler
#sc = StandardScaler()
#X_train = sc.fit_transform(X_train)
#X_tast = sc.fit_transform(X_test)
```

In [ ]:

In [71]:

```
X_train
```

Out[71]:

```
array([[ 148612,       3,       2],
       [ 622500,       5,       2],
       [ 587000,       3,       2],
       ...,
       [ 538888,       5,       2],
       [1920000,       4,       1],
       [ 475000,       3,       1]], dtype=int64)
```

In [72]:

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

Out[72]:

```
LinearRegression()
```

In [73]:

```
Rsquared = regressor.score(X_test, y_test)
```

In [74]:

```
Rsquared
```

Out[74]:

```
1.0
```

In [75]:

```
intercept = regressor.intercept_
intercept
```

Out[75]:

```
-1.1641532182693481e-10
```

In [76]:

```
coefficient = regressor.coef_
coefficient
```

Out[76]:

```
array([ 1.00000000e+00,  1.17313098e-12, -4.85576144e-12])
```

In [77]:

```
#from the above values we can derive formula
# formula for straight line
### y = mx + c
```

new_Price = intercept + -(coefficient)*distancenew_Price = -1.16415221 + -(1.17313098e)* distance

In [79]:
```
y_pred = regressor.predict(X_test)
```

In [80]:
```
y_pred
```

Out[80]:
```
array([289000., 429900., 129000., ..., 985000., 135333., 380000.])
```

In [81]:
```
#these are the predicted house price let compare them to the actual house price
```

In [82]:
```
y_test
```

Out[82]:
```
array([289000, 429900, 129000, ..., 985000, 135333, 380000], dtype=int64)
```

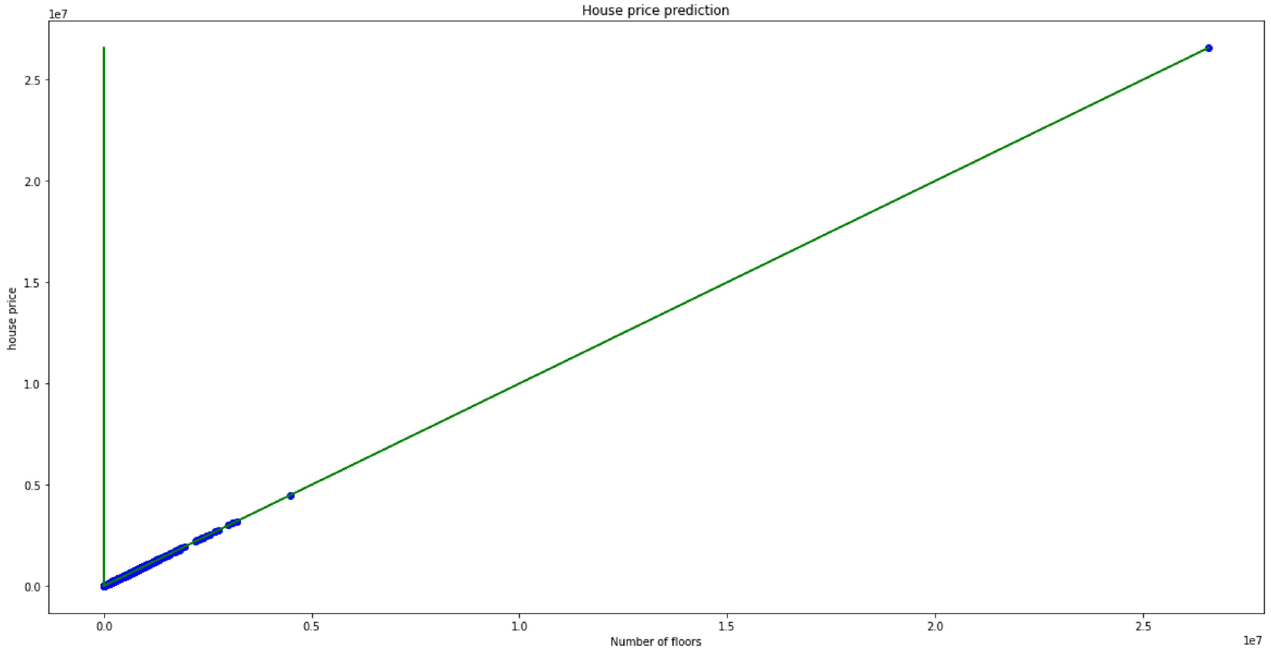
In [83]:
```
y_pred.dtype
```

Out[83]:
```
dtype('float64')
```

In [84]:
```
y_test.dtype
```

Out[84]:
```
dtype('int64')
```

In [102…
```
plt.scatter(y_pred, y_test, color='blue')
plt.plot(X_test, y_pred, color = 'green')
plt.title('House price prediction')
plt.xlabel('Number of floors')
plt.ylabel('house price')
plt.show()
```



new_prediction = regressor.predict([[value]]) print(new_prediction)y_new_prediction = coefficient* 2.5 + intercept print(y_new_pred)You can predict the price of the House using Price = intercept + -(coefficient)*distance(in the above case we found a -ve coefficient) here we can avoid the deployment of Model in dfferent enviornment . Simply using this given formula. we can predict the house price .