Problem statement

predicting the house price in USA.To create a model to help him estimate of what the house would sell for.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df=pd.read_csv("countries")
```

To display top 10 rows

In [3]:
```
1 df.head(10)
```

Out[3]:

	id	name	iso3	iso2	numeric_code	phone_code	capital	currency	currency_name	currency_symbol	tld	native	reg
0	1	Afghanistan	AFG	AF	4	93	Kabul	AFN	Afghan afghani	؋	.af	افغانستان	
1	2	Aland Islands	ALA	AX	248	+358-18	Mariehamn	EUR	Euro	€	.ax	Åland	Eur
2	3	Albania	ALB	AL	8	355	Tirana	ALL	Albanian lek	Lek	.al	Shqipëria	Eur
3	4	Algeria	DZA	DZ	12	213	Algiers	DZD	Algerian dinar	دج	.dz	الجزائر	A
4	5	American Samoa	ASM	AS	16	+1-684	Pago Pago	USD	US Dollar	$	.as	American Samoa	Oce
5	6	Andorra	AND	AD	20	376	Andorra la Vella	EUR	Euro	€	.ad	Andorra	Eur
6	7	Angola	AGO	AO	24	244	Luanda	AOA	Angolan kwanza	Kz	.ao	Angola	A
7	8	Anguilla	AIA	AI	660	+1-264	The Valley	XCD	East Caribbean dollar	$	.ai	Anguilla	Amer
8	9	Antarctica	ATA	AQ	10	672	NaN	AAD	Antarctican dollar	$	.aq	Antarctica	F
9	10	Antigua And Barbuda	ATG	AG	28	+1-268	St. John's	XCD	Eastern Caribbean dollar	$	.ag	Antigua and Barbuda	Amer

Data Cleaning And Pre-Processing

In [4]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 250 entries, 0 to 249
Data columns (total 19 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   id               250 non-null    int64  
 1   name             250 non-null    object 
 2   iso3             250 non-null    object 
 3   iso2             249 non-null    object 
 4   numeric_code     250 non-null    int64  
 5   phone_code       250 non-null    object 
 6   capital          245 non-null    object 
 7   currency         250 non-null    object 
 8   currency_name    250 non-null    object 
 9   currency_symbol  250 non-null    object 
 10  tld              250 non-null    object 
 11  native           249 non-null    object 
 12  region           248 non-null    object 
 13  subregion        247 non-null    object 
 14  timezones        250 non-null    object 
 15  latitude         250 non-null    float64
 16  longitude        250 non-null    float64
 17  emoji            250 non-null    object 
 18  emojiU           250 non-null    object 
dtypes: float64(2), int64(2), object(15)
memory usage: 37.2+ KB
```

In [5]:
```
# Display the statistical summary
df.describe()
```

Out[5]:

	id	numeric_code	latitude	longitude
count	250.000000	250.00000	250.000000	250.00000
mean	125.500000	435.80400	16.402597	13.52387
std	72.312977	254.38354	26.757204	73.45152
min	1.000000	4.00000	-74.650000	-176.20000
25%	63.250000	219.00000	1.000000	-49.75000
50%	125.500000	436.00000	16.083333	17.00000
75%	187.750000	653.50000	39.000000	48.75000
max	250.000000	926.00000	78.000000	178.00000

In [6]:
```
# To display the col headings
df.columns
```

Out[6]:
```
Index(['id', 'name', 'iso3', 'iso2', 'numeric_code', 'phone_code', 'capital',
       'currency', 'currency_name', 'currency_symbol', 'tld', 'native',
       'region', 'subregion', 'timezones', 'latitude', 'longitude', 'emoji',
       'emojiU'],
      dtype='object')
```

In [7]:
```
cols=df.dropna(axis=1)
```

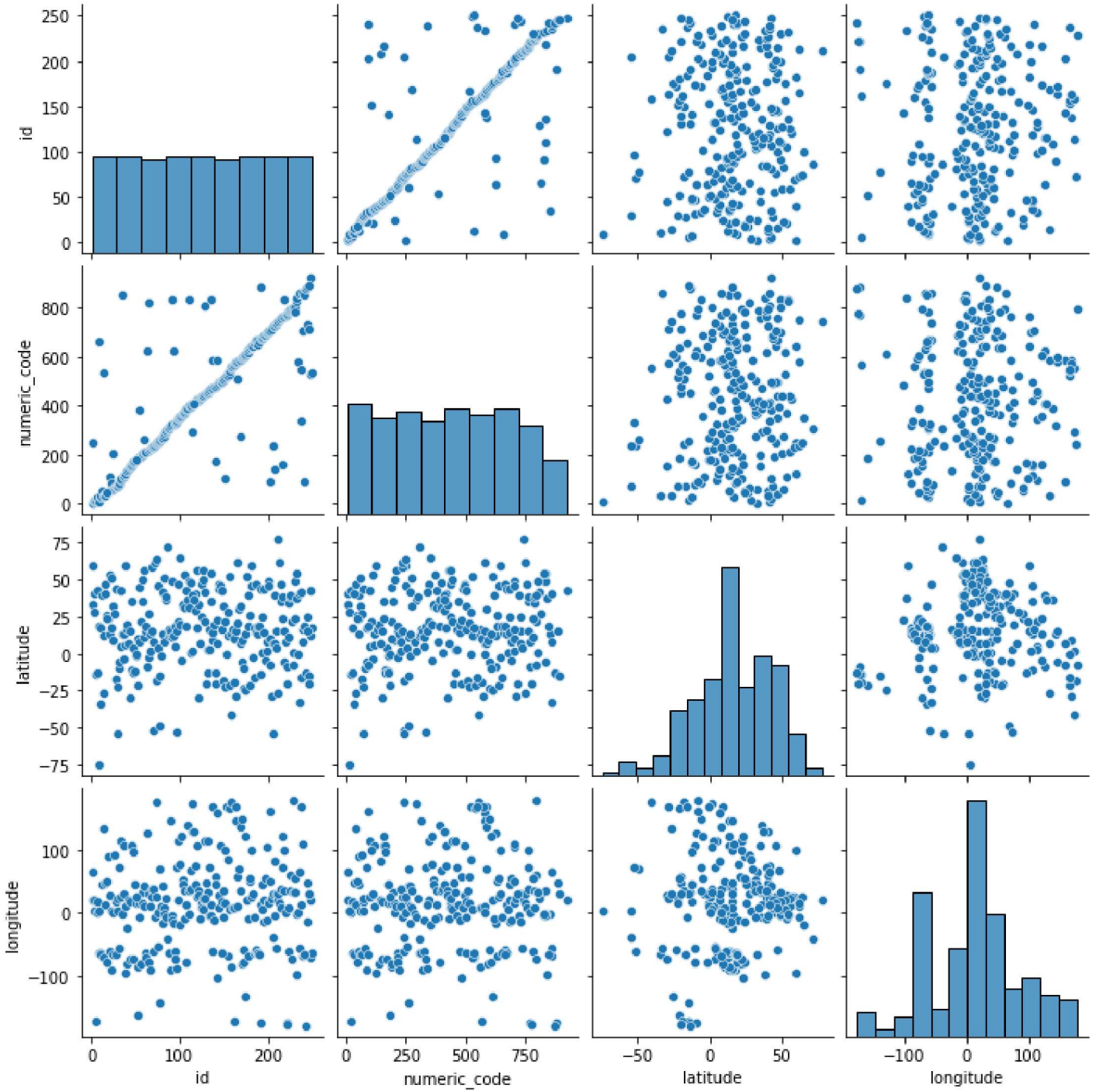
In [8]:
```
cols.columns
```

Out[8]:
```
Index(['id', 'name', 'iso3', 'numeric_code', 'phone_code', 'currency',
       'currency_name', 'currency_symbol', 'tld', 'timezones', 'latitude',
       'longitude', 'emoji', 'emojiU'],
      dtype='object')
```

EDA and Visualization

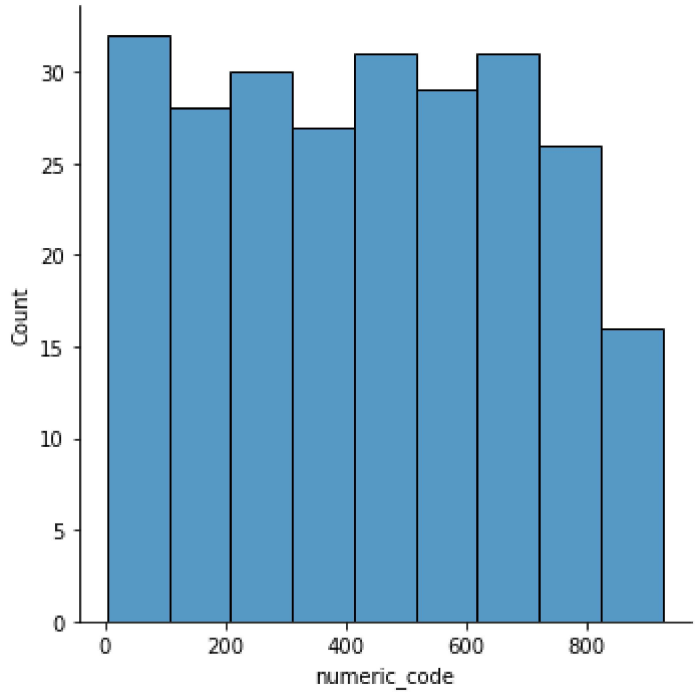
In [9]:
```
1 sns.pairplot(cols)
```

Out[9]:
```
<seaborn.axisgrid.PairGrid at 0x1f0bbbe6040>
```


id
numeric_code
latitude
longitude
250
200
150
100
50
0
800
600
400
200
0
75
50
25
0
−25
−50
−75
100
0
−100
0
100
200
0
250
500
750
−50
0
50
−100
0
100

In [11]:
```
1 sns.displot(df['numeric_code'])
```

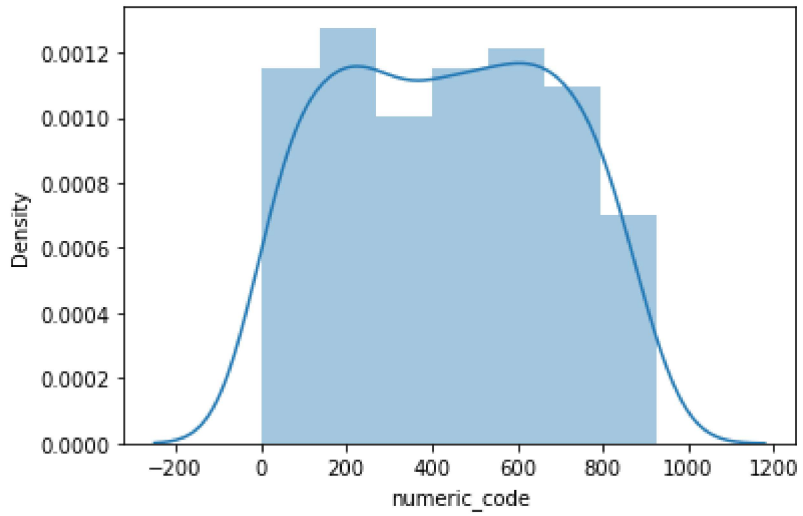
Out[11]: <seaborn.axisgrid.FacetGrid at 0x1f0baa968b0>



In [12]:
```
# We use displot in older version we get distplot use displot
sns.distplot(df['numeric_code'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a dep
recated function and will be removed in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[12]: <AxesSubplot:xlabel='numeric_code', ylabel='Density'>



In [13]:

```
df1=cols[['id', 'numeric_code',
       'latitude', 'longitude']]
df1
```

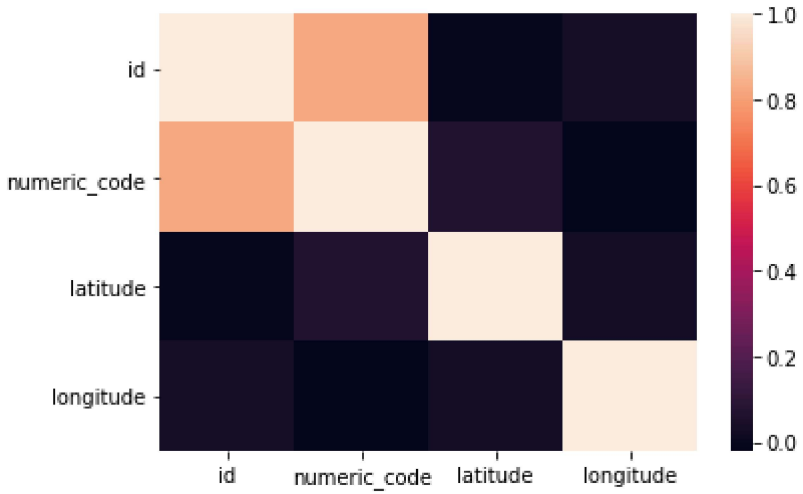
Out[13]:

	id	numeric_code	latitude	longitude
0	1	4	33.000000	65.0
1	2	248	60.116667	19.9
2	3	8	41.000000	20.0
3	4	12	28.000000	3.0
4	5	16	-14.333333	-170.0
...	...	...	...	...
245	243	876	-13.300000	-176.2
246	244	732	24.500000	-13.0
247	245	887	15.000000	48.0
248	246	894	-15.000000	30.0
249	247	716	-20.000000	30.0

250 rows × 4 columns

In [14]:
```
1 sns.heatmap(df1.corr())
```

Out[14]: <AxesSubplot:>



To train the model - MODEL BUILD

Going to train linear regression model;We split our data into 2 variables x and y where x is independent var(input) and y is dependent on x(output), we could ignore address col as it is not required for our model

In [15]:
```
1 x=df1[['id', 'numeric_code',
2        'latitude', 'longitude']]
3 y=df1[['id']]
```

To split the dataset into test data

In [16]:
```
1 # importing lib for splitting test data
2 from sklearn.model_selection import train_test_split
```

In [17]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```

In [18]:
```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[18]:
```
LinearRegression()
```

In [19]:
```
print(lr.intercept_)
```

```
[-4.26325641e-14]
```

In [20]:
```
print(lr.score(x_test,y_test))
```

```
1.0
```

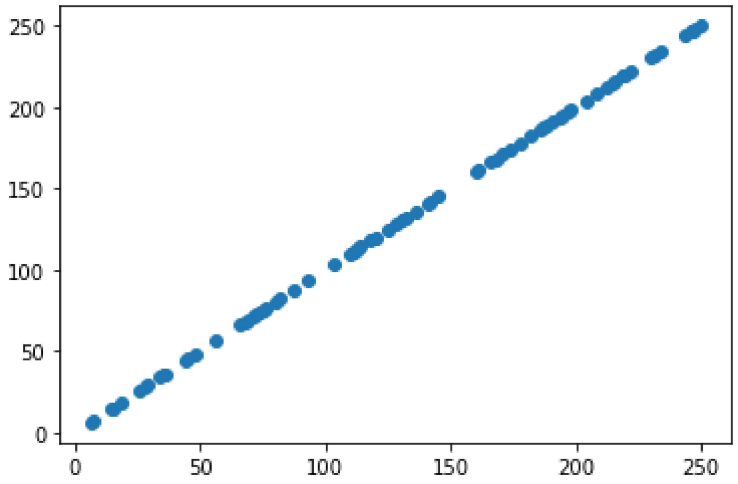
In [21]:
```
coeff=pd.DataFrame(lr.coef_)
coeff
```

Out[21]:

	0	1	2	3
0	1.0	4.287949e-17	-5.979602e-18	-6.902071e-17

In [22]:
```
1 pred = lr.predict(x_test)
2 plt.scatter(y_test,pred)
```

Out[22]: <matplotlib.collections.PathCollection at 0x1f0bd9bf100>



In [23]:
```
1 from sklearn.linear_model import Ridge,Lasso
```

In [24]:
```
1 rr=Ridge(alpha=10)
2 rr.fit(x_train,y_train)
```

Out[24]: Ridge(alpha=10)

In [25]:
```
1 rr.score(x_test,y_test)
```

Out[25]: 0.9999999995085121

In [26]:
```
1 la=Lasso(alpha=10)
2 la.fit(x_train,y_train)
```

Out[26]: Lasso(alpha=10)

In [27]:
```
1 la.score(x_test,y_test)
```

Out[27]: 0.9999919038869991

ELASTIC NET

In [28]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[28]:
```
ElasticNet()
```

In [29]:
```
print(en.coef_)
```

```
[ 9.99616424e-01  8.43641639e-05 -0.00000000e+00  0.00000000e+00]
```

In [30]:
```
print(en.intercept_)
```

```
[0.01098077]
```

In [31]:
```
prediction=en.predict(x_test)
prediction
```

Out[31]:
```
array([113.00171976, 144.99686936, 177.9958536 , 191.01212688,
       136.02908973,  48.00691101, 233.97023948, 165.99050154,
       120.00105947, 194.99405719,  74.0033497 , 131.99898748,
        29.00593128, 118.00089861, 215.94145781, 207.94334532,
        26.00640709, 160.99697509, 203.99262975, 181.99541603,
       112.00159715, 221.99129341,  35.06960259,  68.00447006,
         7.01032048,  36.00628335, 103.00218096,  80.00307299,
       193.99410331, 231.99167585, 170.99583898, 214.99262861,
       247.99397504, 197.99358138,  69.00417085,   6.0103666 ,
       187.9953924 , 218.99143177, 245.99204254, 159.99702121,
        34.00637559,  93.02795141,  15.00860169, 127.99950941,
        75.00330358, 185.99548464, 111.00214946, 140.97166023,
       141.99734518,  82.00281202,  87.00359379,  18.00812588,
        44.00692676, 196.9936275 ,  72.00310449, 124.99998523,
       246.97664215, 211.99276697,  28.00614612, 113.99222486,
       110.03897835, 130.9990336 , 243.9791427 , 167.99589298,
        56.00569841, 229.99058699,  14.00864781,  71.00382552,
        66.00439357, 219.99138565, 173.99553189, 249.96013714,
       186.99543852,  76.00325746,  45.00688064])
```

In [32]:
```
print(en.score(x_test,y_test))
```

```
0.9999999464811962
```

EVALUATION METRICS

In [33]:
```
from sklearn import metrics
```

In [34]:
```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error: 0.010114911438299632
```

In [35]:
```
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error: 0.0002817255140920051
```

In [36]:
```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
Root Mean Squared Error: 0.016784680935067103
```

MODEL SAVING

In [37]:
```
import pickle
```

In [38]:
```
filename='prediction4'
pickle.dump(lr,open(filename,'wb'))
```

In [ ]:
```

```