31-07-2023

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

In [436]:
```
a=pd.read_csv(r"C:\Users\user\Downloads\20_states.csv")
a
```

Out[436]:

	id	name	country_id	country_code	country_name	state_code	type	latitude	longitude
0	3901	Badakhshan	1	AF	Afghanistan	BDS	NaN	36.734772	70.811995
1	3871	Badghis	1	AF	Afghanistan	BDG	NaN	35.167134	63.769538
2	3875	Baghlan	1	AF	Afghanistan	BGL	NaN	36.178903	68.745306
3	3884	Balkh	1	AF	Afghanistan	BAL	NaN	36.755060	66.897537
4	3872	Bamyan	1	AF	Afghanistan	BAM	NaN	34.810007	67.821210
...	...	...	...	...	...	...	...	...	...
5072	1953	Mashonaland West Province	247	ZW	Zimbabwe	MW	NaN	-17.485103	29.788925
5073	1960	Masvingo Province	247	ZW	Zimbabwe	MV	NaN	-20.624151	31.262637
5074	1954	Matabeleland North Province	247	ZW	Zimbabwe	MN	NaN	-18.533157	27.549585
5075	1952	Matabeleland South Province	247	ZW	Zimbabwe	MS	NaN	-21.052337	29.045993
5076	1957	Midlands Province	247	ZW	Zimbabwe	MI	NaN	-19.055201	29.603549

5077 rows × 9 columns

In [437]:
```
a=a.head(10)
a
```

Out[437]:

	id	name	country_id	country_code	country_name	state_code	type	latitude	longitude
0	3901	Badakhshan	1	AF	Afghanistan	BDS	NaN	36.734772	70.811995
1	3871	Badghis	1	AF	Afghanistan	BDG	NaN	35.167134	63.769538
2	3875	Baghlan	1	AF	Afghanistan	BGL	NaN	36.178903	68.745306
3	3884	Balkh	1	AF	Afghanistan	BAL	NaN	36.755060	66.897537
4	3872	Bamyan	1	AF	Afghanistan	BAM	NaN	34.810007	67.821210
5	3892	Daykundi	1	AF	Afghanistan	DAY	NaN	33.669495	66.046353
6	3899	Farah	1	AF	Afghanistan	FRA	NaN	32.495328	62.262663
7	3889	Faryab	1	AF	Afghanistan	FYB	NaN	36.079561	64.905955
8	3870	Ghazni	1	AF	Afghanistan	GHA	NaN	33.545059	68.417397
9	3888	Ghōr	1	AF	Afghanistan	GHO	NaN	34.099578	64.905955

In [438]:
```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   id            10 non-null     int64  
 1   name          10 non-null     object 
 2   country_id    10 non-null     int64  
 3   country_code  10 non-null     object 
 4   country_name  10 non-null     object 
 5   state_code    10 non-null     object 
 6   type          0 non-null      object 
 7   latitude      10 non-null     float64
 8   longitude     10 non-null     float64
dtypes: float64(2), int64(2), object(5)
memory usage: 848.0+ bytes
```

In [439]:
```
a.columns
```

Out[439]:
```
Index(['id', 'name', 'country_id', 'country_code', 'country_name',
       'state_code', 'type', 'latitude', 'longitude'],
      dtype='object')
```

In [440]:
```
d=a[['id', 'name', 'country_id', 'country_code', 'country_name',
       'state_code', 'type', 'latitude', 'longitude']]
d
```

Out[440]:

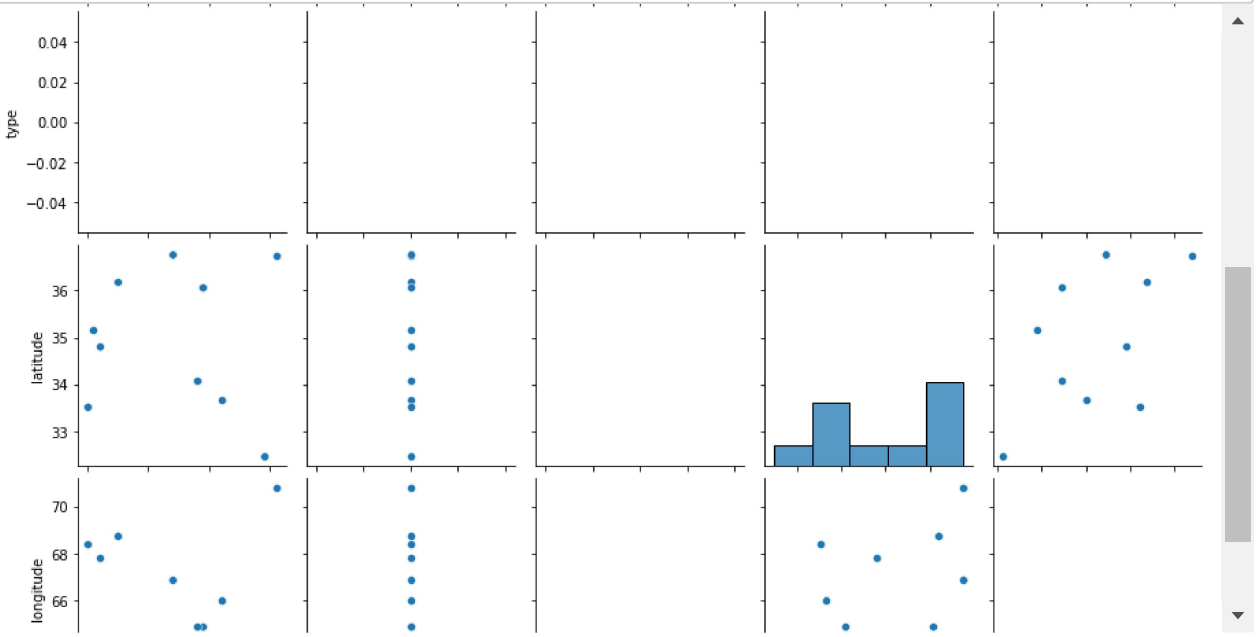
	id	name	country_id	country_code	country_name	state_code	type	latitude	longitude
0	3901	Badakhshan	1	AF	Afghanistan	BDS	NaN	36.734772	70.811995
1	3871	Badghis	1	AF	Afghanistan	BDG	NaN	35.167134	63.769538
2	3875	Baghlan	1	AF	Afghanistan	BGL	NaN	36.178903	68.745306
3	3884	Balkh	1	AF	Afghanistan	BAL	NaN	36.755060	66.897537
4	3872	Bamyan	1	AF	Afghanistan	BAM	NaN	34.810007	67.821210
5	3892	Daykundi	1	AF	Afghanistan	DAY	NaN	33.669495	66.046353
6	3899	Farah	1	AF	Afghanistan	FRA	NaN	32.495328	62.262663
7	3889	Faryab	1	AF	Afghanistan	FYB	NaN	36.079561	64.905955
8	3870	Ghazni	1	AF	Afghanistan	GHA	NaN	33.545059	68.417397
9	3888	Ghōr	1	AF	Afghanistan	GHO	NaN	34.099578	64.905955

In [441]:
```
d.describe()
```

Out[441]:

	id	country_id	latitude	longitude
count	10.000000	10.0	10.000000	10.000000
mean	3884.100000	1.0	34.953490	66.458391
std	11.589746	0.0	1.477933	2.579742
min	3870.000000	1.0	32.495328	62.262663
25%	3872.750000	1.0	33.777016	64.905955
50%	3886.000000	1.0	34.988570	66.471945
75%	3891.250000	1.0	36.154067	68.268350
max	3901.000000	1.0	36.755060	70.811995

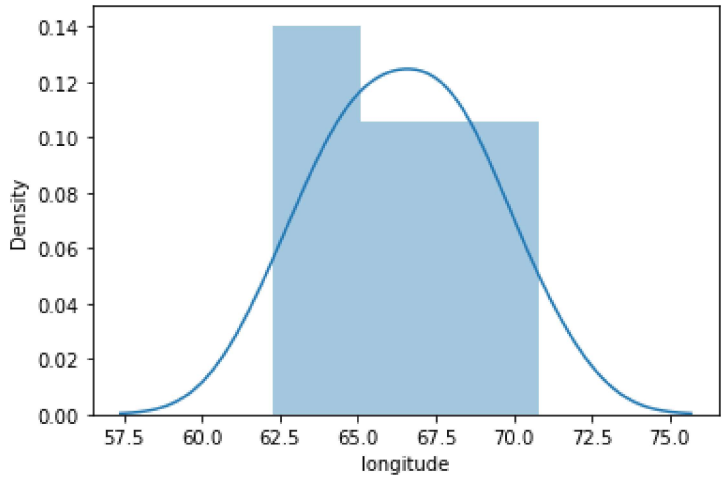
In [442]:
```
sns.pairplot(d)
```



In [443]:
```
sns.distplot(a['longitude'])
```

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

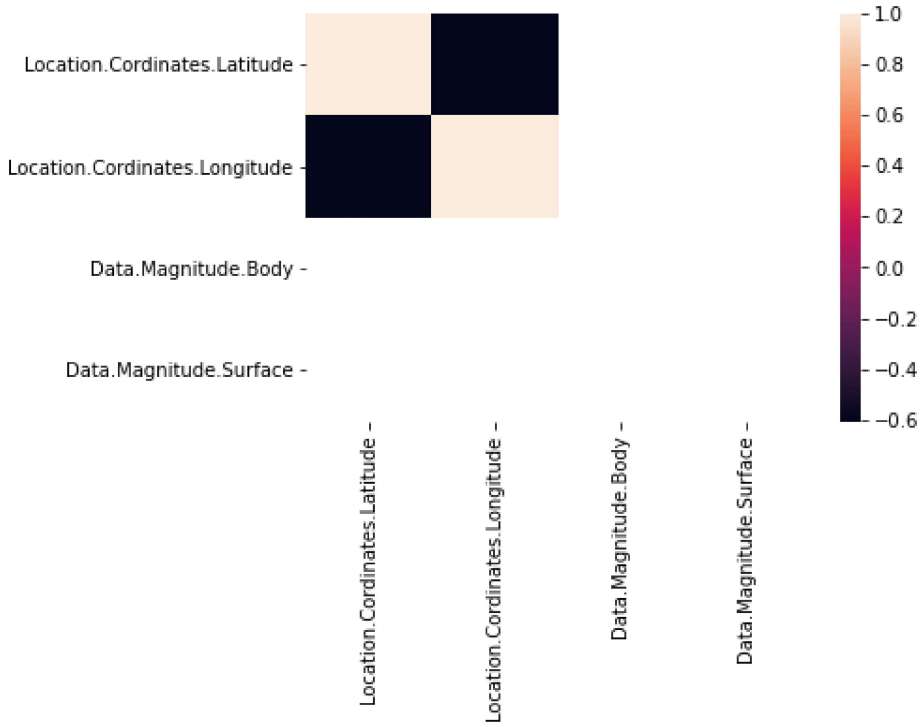
Out[443]: <AxesSubplot:xlabel='longitude', ylabel='Density'>



In [404]:
```
x1=a[['id',  'country_id', 'latitude']]
```

In [405]:
```
sns.heatmap(x1.corr())
```

Out[405]: <AxesSubplot:>



In [444]:
```
x=a[['id',  'country_id', 'latitude']]
y=a['longitude']
```

In [445]:
```
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)
```

In [446]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[446]: LinearRegression()

In [447]:
```
print(lr.intercept_)
```

```
550.0605887546454
```

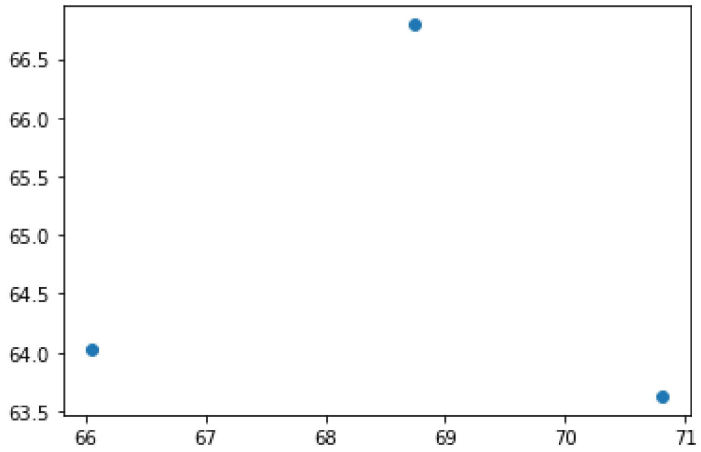
In [448]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[448]:

	Co-efficient
id	-0.126970
country_id	0.000000
latitude	0.241712

In [449]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[449]:
```
<matplotlib.collections.PathCollection at 0x190c6452130>
```



In [450]:
```
print(lr.score(x_test,y_test))
```

```
-4.207250283337438
```

In [451]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [452]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[452]:
```
Ridge(alpha=10)
```

In [453]:
```
rr.score(x_test,y_test)
```

Out[453]:
```
-4.528670035890717
```

In [454]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[454]:
```
Lasso(alpha=10)
```

In [455]:
```
la.score(x_test,y_test)
```

Out[455]:
```
-2.9515925374866674
```

In [456]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[456]:
```
ElasticNet()
```

In [457]:
```
print(en.coef_)
```

```
[-0.12806125  0.          0.        ]
```

In [458]:
```
print(en.intercept_)
```

```
562.6840904609145
```

In [459]:
```
print(en.predict(x_test))
```

```
[63.11714975 66.44674228 64.26970101]
```

In [460]:
```
en.score(x_test,y_test)
```

Out[460]:
```
-4.922673415078662
```

In [461]:
```
from sklearn import metrics
```

In [462]:
```
print("Mean Absolute Error",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error 3.716914720062898
```

In [463]:
```
print("Mean Squared Error",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error 19.826254267954145
```

In [464]:
```
print(" Root Mean Squared Error",np.sqrt(metrics.mean_squared_error(y_test,prediction))
```

```
 Root Mean Squared Error 4.452668218939532
```

In [465]:
```
import pickle
```

In [466]:
```
filename="prediction"
pickle.dump(lr,open(filename,'wb'))
```

In [467]:
```
import pandas as pd
import pickle
```

In [468]:
```
filename="prediction"
model=pickle.load(open(filename,"rb"))
```

In [469]:
```
real=[[10,20,24,],[15,30,36]]
result=model.predict(real)
```

In [470]:
```
result
```

Out[470]:
```
array([554.59196495, 556.85765305])
```