DAY 10:

Cities Dataset

In [1]:

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

In [2]:

```
df=pd.read_csv(r"C:\Users\user\Downloads\21_cities.csv")[0:100]
df
```

Out[2]:

	id	name	state_id	state_code	state_name	country_id	country_code	country_nar
0	52	Ashkāsham	3901	BDS	Badakhshan	1	AF	Afghanist
1	68	Fayzabad	3901	BDS	Badakhshan	1	AF	Afghanist
2	78	Jurm	3901	BDS	Badakhshan	1	AF	Afghanist
3	84	Khandūd	3901	BDS	Badakhshan	1	AF	Afghanist
4	115	Rāghistān	3901	BDS	Badakhshan	1	AF	Afghanist
...	...	...	...	...	...	...	...	
95	180	Bashkia Poliçan	629	BR	Berat District	3	AL	Albar
96	186	Bashkia Skrapar	629	BR	Berat District	3	AL	Albar
97	191	Berat	629	BR	Berat District	3	AL	Albar
98	280	Çorovodë	629	BR	Berat District	3	AL	Albar
99	219	Kuçovë	629	BR	Berat District	3	AL	Albar

100 rows × 11 columns

In [3]:

```
df.fillna(value=1)
```

Out[3]:

	id	name	state_id	state_code	state_name	country_id	country_code	country_nar
0	52	Ashkāsham	3901	BDS	Badakhshan	1	AF	Afghanist
1	68	Fayzabad	3901	BDS	Badakhshan	1	AF	Afghanist
2	78	Jurm	3901	BDS	Badakhshan	1	AF	Afghanist
3	84	Khandūd	3901	BDS	Badakhshan	1	AF	Afghanist
4	115	Rāghistān	3901	BDS	Badakhshan	1	AF	Afghanist
...	...	...	...	...	...	...	...	
95	180	Bashkia Poliçan	629	BR	Berat District	3	AL	Alba
96	186	Bashkia Skrapar	629	BR	Berat District	3	AL	Alba
97	191	Berat	629	BR	Berat District	3	AL	Alba
98	280	Çorovodë	629	BR	Berat District	3	AL	Alba
99	219	Kuçovë	629	BR	Berat District	3	AL	Alba

100 rows × 11 columns

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   id            100 non-null    int64  
 1   name          100 non-null    object 
 2   state_id      100 non-null    int64  
 3   state_code    100 non-null    object 
 4   state_name    100 non-null    object 
 5   country_id    100 non-null    int64  
 6   country_code  100 non-null    object 
 7   country_name  100 non-null    object 
 8   latitude      100 non-null    float64
 9   longitude     100 non-null    float64
 10  wikiDataId    100 non-null    object 
dtypes: float64(2), int64(3), object(6)
memory usage: 8.7+ KB
```

In [5]:

```
df.columns
```

Out[5]:

```
Index(['id', 'name', 'state_id', 'state_code', 'state_name', 'country_id',
       'country_code', 'country_name', 'latitude', 'longitude', 'wikiDataI
d'],
      dtype='object')
```

Linear Regression

In [6]:

```
x=df[['id','state_id','country_id', 'latitude']]
y=df[ 'longitude']
```

In [7]:

```
# to split my dataset into test and train data
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 [8]:

```
from sklearn.linear_model import LinearRegression

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

Out[8]:

```
LinearRegression()
```

In [9]:

```
print(lr.score(x_test,y_test))
```

```
0.9516406711634929
```

In [10]:

```
lr.score(x_train,y_train)
```

Out[10]:

```
0.9606693693910433
```

Ridge Regression

In [11]:

```
from sklearn.linear_model import Ridge,Lasso
```

In [12]:

```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_test,y_test)
```

Out[12]:

```
0.9519373388694963
```

Lasso Regression

In [13]:

```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[13]:

```
Lasso(alpha=10)
```

In [14]:

```
la.score(x_test,y_test)
```

Out[14]:

```
0.9495766756365002
```

Elastic regression

In [15]:

```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[15]:

```
ElasticNet()
```

In [16]:

```
print(en.intercept_)
```

```
-13.42210774825324
```

In [17]:

```
predict=(en.predict(x_test))
```

In [18]:

```
print(en.score(x_test,y_test))
```

```
0.9533592794479002
```

Evalution matrics

In [19]:

```
from sklearn import metrics
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,predict))
```

```
Mean Absolute Error: 2.2167004542532402
```

In [20]:

```
print("Mean Square Error:",metrics.mean_squared_error(y_test,predict))
```

```
Mean Square Error: 6.906591121925426
```

In [21]:

```
print("Root Mean Square Error:",np.sqrt(metrics.mean_squared_error(y_test,predict)))
```

```
Root Mean Square Error: 2.6280394064635764
```

In [22]:

```
import pickle
```

In [23]:

```
filename="predict"
```

In [24]:

```
pickle.dump(lr,open(filename,'wb'))
```

In [ ]:

```

```