# DAY 10:

# States Dataset

In [1]:

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

In [14]:

```
df=pd.read_csv(r"C:\Users\user\Downloads\20_states.csv")[0:100]
df
```

Out[14]:

| | id | name | country_id | country_code | country_name | state_code | type | latitude |
|---|---|---|---|---|---|---|---|---|
| **0** | 3901 | Badakhshan | 1 | AF | Afghanistan | BDS | NaN | 36.734772 |
| **1** | 3871 | Badghis | 1 | AF | Afghanistan | BDG | NaN | 35.167134 |
| **2** | 3875 | Baghlan | 1 | AF | Afghanistan | BGL | NaN | 36.178903 |
| **3** | 3884 | Balkh | 1 | AF | Afghanistan | BAL | NaN | 36.755060 |
| **4** | 3872 | Bamyan | 1 | AF | Afghanistan | BAM | NaN | 34.810007 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **95** | 1105 | Chlef | 4 | DZ | Algeria | 02 | NaN | 36.169351 |
| **96** | 1121 | Constantine | 4 | DZ | Algeria | 25 | NaN | 36.337391 |
| **97** | 4912 | Djanet | 4 | DZ | Algeria | 56 | NaN | 23.831087 |
| **98** | 1098 | Djelfa | 4 | DZ | Algeria | 17 | NaN | 34.670396 |
| **99** | 1129 | El Bayadh | 4 | DZ | Algeria | 32 | NaN | 32.714882 |

100 rows × 9 columns

In [15]:

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

Out[15]:

| | id | name | country_id | country_code | country_name | state_code | type | latitude |
|---|---|---|---|---|---|---|---|---|
| **0** | 3901 | Badakhshan | 1 | AF | Afghanistan | BDS | 1 | 36.734772 |
| **1** | 3871 | Badghis | 1 | AF | Afghanistan | BDG | 1 | 35.167134 |
| **2** | 3875 | Baghlan | 1 | AF | Afghanistan | BGL | 1 | 36.178903 |
| **3** | 3884 | Balkh | 1 | AF | Afghanistan | BAL | 1 | 36.755060 |
| **4** | 3872 | Bamyan | 1 | AF | Afghanistan | BAM | 1 | 34.810007 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **95** | 1105 | Chlef | 4 | DZ | Algeria | 02 | 1 | 36.169351 |
| **96** | 1121 | Constantine | 4 | DZ | Algeria | 25 | 1 | 36.337391 |
| **97** | 4912 | Djanet | 4 | DZ | Algeria | 56 | 1 | 23.831087 |
| **98** | 1098 | Djelfa | 4 | DZ | Algeria | 17 | 1 | 34.670396 |
| **99** | 1129 | El Bayadh | 4 | DZ | Algeria | 32 | 1 | 32.714882 |

100 rows × 9 columns

In [16]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   id            100 non-null    int64
 1   name          100 non-null    object
 2   country_id    100 non-null    int64
 3   country_code  100 non-null    object
 4   country_name  100 non-null    object
 5   state_code    100 non-null    object
 6   type          0 non-null      object
 7   latitude      100 non-null    float64
 8   longitude     100 non-null    float64
dtypes: float64(2), int64(2), object(5)
memory usage: 7.2+ KB
```

In [10]:

```
df.columns
```

Out[10]:

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

## Linear Regression

In [18]:

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

In [19]:

```
# 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 [20]:

```
from sklearn.linear_model import LinearRegression

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

Out[20]:

```
LinearRegression()
```

In [21]:

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

```
0.6521172868477666
```

In [22]:

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

Out[22]:

```
0.48303110990674214
```

## Ridge Regression

In [23]:

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

In [24]:

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

Out[24]:

```
0.7649291266346778
```

## Lasso Regression

In [25]:

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

Out[25]:

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

In [26]:

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

Out[26]:

```
0.6211149043890036
```

## Elastic regression

In [27]:

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

Out[27]:

```
ElasticNet()
```

In [28]:

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

```
78.3803372684851
```

In [29]:

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

In [30]:

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

```
0.7500155655941771
```

## Evalution matrics

In [31]:

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

```
Mean Absolute Error: 10.53639869536998
```

In [32]:

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

```
Mean Square Error: 157.71330015496278
```

In [33]:

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

```
Root Mean Square Error: 12.558395604334288
```

In [34]:

```
import pickle
```

In [35]:

```
filename="predict"
```

In [36]:

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

In [ ]:

```
```