wjafkibqs

July 31, 2023

[12]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[13]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[14]:
```
df=pd.read_csv("/content/drive/MyDrive/mydatasets/21_cities.csv")
df
```

[14]:
```
            id                 name  state_id state_code        state_name  \
0           52            Ashkāsham      3901        BDS        Badakhshan
1           68             Fayzabad      3901        BDS        Badakhshan
2           78                 Jurm      3901        BDS        Badakhshan
3           84              Khandūd      3901        BDS        Badakhshan
4          115            Rāghistān      3901        BDS        Badakhshan
…          …                    …         …          …                 …
150449  131496             Redcliff      1957         MI  Midlands Province
150450  131502             Shangani      1957         MI  Midlands Province
150451  131503             Shurugwi      1957         MI  Midlands Province
150452  131504    Shurugwi District      1957         MI  Midlands Province
150453  131508  Zvishavane District      1957         MI  Midlands Province

        country_id country_code country_name  latitude  longitude wikiDataId
0                1           AF  Afghanistan  36.68333   71.53333   Q4805192
1                1           AF  Afghanistan  37.11664   70.58002    Q156558
2                1           AF  Afghanistan  36.86477   70.83421  Q10308323
3                1           AF  Afghanistan  36.95127   72.31800   Q3290334
4                1           AF  Afghanistan  37.66079   70.67346   Q2670909
…              …          …           …         …        …          …
150449         247           ZW     Zimbabwe -19.03333   29.78333    Q584001
150450         247           ZW     Zimbabwe -19.78333   29.36667  Q32017959
150451         247           ZW     Zimbabwe -19.67016   30.00589  Q32019023
150452         247           ZW     Zimbabwe -19.75000   30.16667   Q7505444
```

```
150453         247           ZW     Zimbabwe -20.30345   30.07514  Q24235929

[150454 rows x 11 columns]
```

[15]:
```python
df.head()
```

[15]:
```
    id       name  state_id state_code  state_name  country_id country_code  \
0   52  Ashkāsham      3901        BDS  Badakhshan           1           AF
1   68   Fayzabad      3901        BDS  Badakhshan           1           AF
2   78       Jurm      3901        BDS  Badakhshan           1           AF
3   84    Khandūd      3901        BDS  Badakhshan           1           AF
4  115  Rāghistān      3901        BDS  Badakhshan           1           AF

  country_name  latitude  longitude wikiDataId
0  Afghanistan  36.68333   71.53333   Q4805192
1  Afghanistan  37.11664   70.58002    Q156558
2  Afghanistan  36.86477   70.83421  Q10308323
3  Afghanistan  36.95127   72.31800   Q3290334
4  Afghanistan  37.66079   70.67346   Q2670909
```

1 Data Cleaning and Data Preprocessing

[16]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150454 entries, 0 to 150453
Data columns (total 11 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   id            150454 non-null  int64
 1   name          150454 non-null  object
 2   state_id      150454 non-null  int64
 3   state_code    150129 non-null  object
 4   state_name    150454 non-null  object
 5   country_id    150454 non-null  int64
 6   country_code  150406 non-null  object
 7   country_name  150454 non-null  object
 8   latitude      150454 non-null  float64
 9   longitude     150454 non-null  float64
 10  wikiDataId    147198 non-null  object
dtypes: float64(2), int64(3), object(6)
memory usage: 12.6+ MB
```

[17]:
```python
df.describe()
```

```
[17]:                   id       state_id     country_id       latitude  \
       count  150454.000000  150454.000000  150454.000000  150454.000000
       mean    76407.091689    2678.377677     140.658460      31.556175
       std     44357.755335    1363.513591      70.666123      22.813220
       min         1.000000       1.000000       1.000000     -75.000000
       25%     38160.250000    1451.000000      82.000000      19.000000
       50%     75975.500000    2174.000000     142.000000      40.684720
       75%    115204.750000    3905.000000     207.000000      47.239220
       max    153528.000000    5116.000000     247.000000      73.508190

                  longitude
       count  150454.000000
       mean        2.369557
       std        68.012770
       min      -179.121980
       25%       -58.468150
       50%         8.669980
       75%        27.750000
       max       179.466000
```

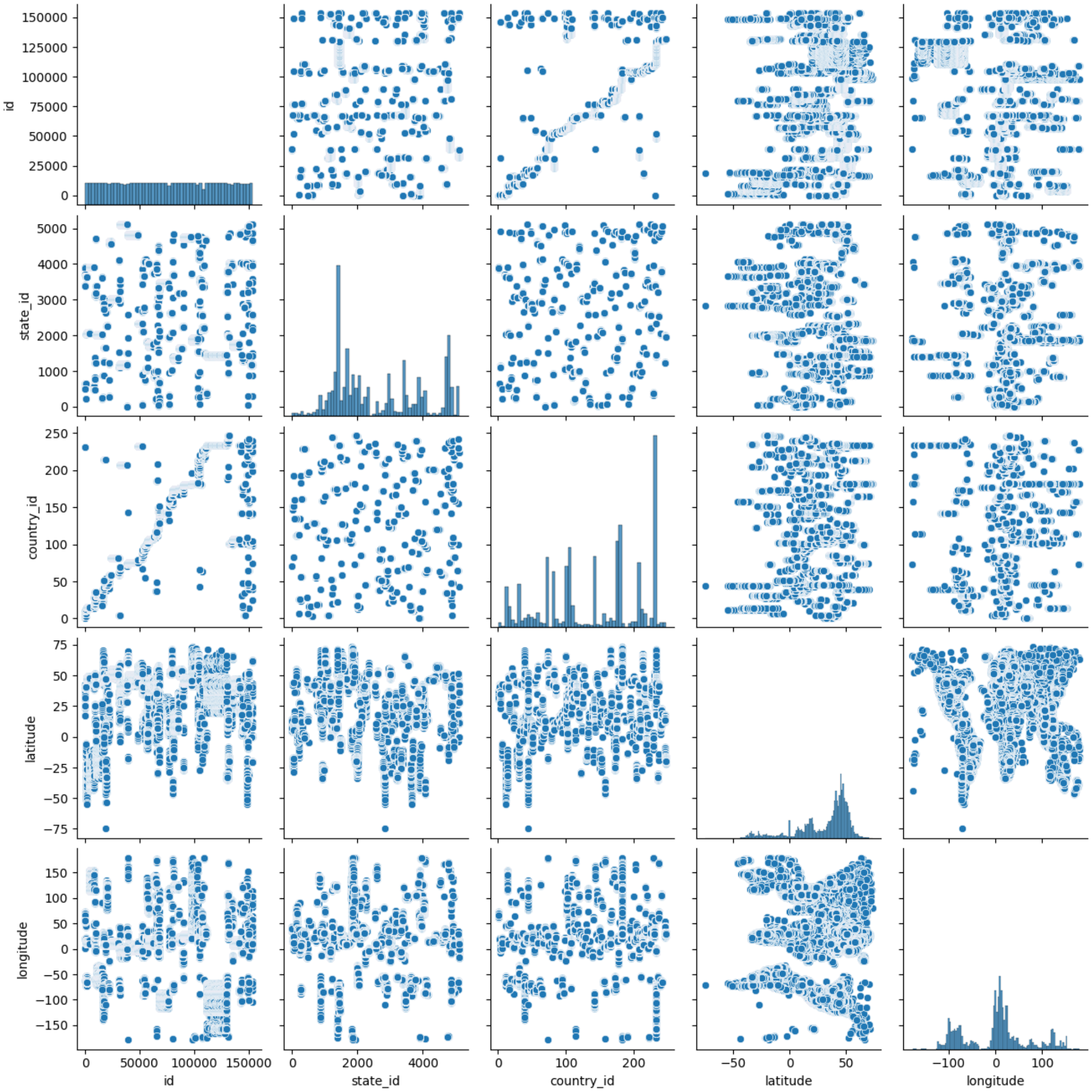
```
[18]: df.columns
```

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

2 EDA and Visualization

```
[19]: sns.pairplot(df)
```

```
[19]: <seaborn.axisgrid.PairGrid at 0x7eea24d36050>
```

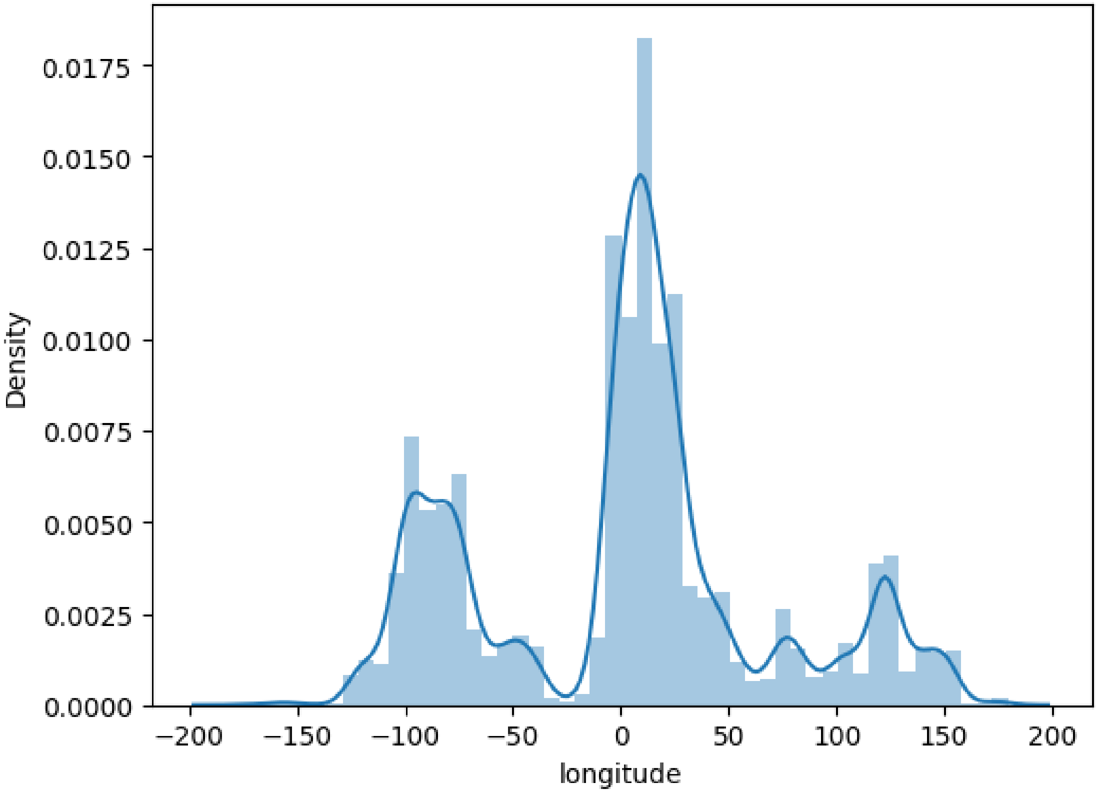


[20]:

```
sns.distplot(df['longitude'])
```

```
<ipython-input-20-4c5c6f107715>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['longitude'])
```

```
[20]: <Axes: xlabel='longitude', ylabel='Density'>
```



```
[21]: df1=df[['id', 'state_id', 'country_id',
             'latitude', 'longitude']].dropna()
      df1
```

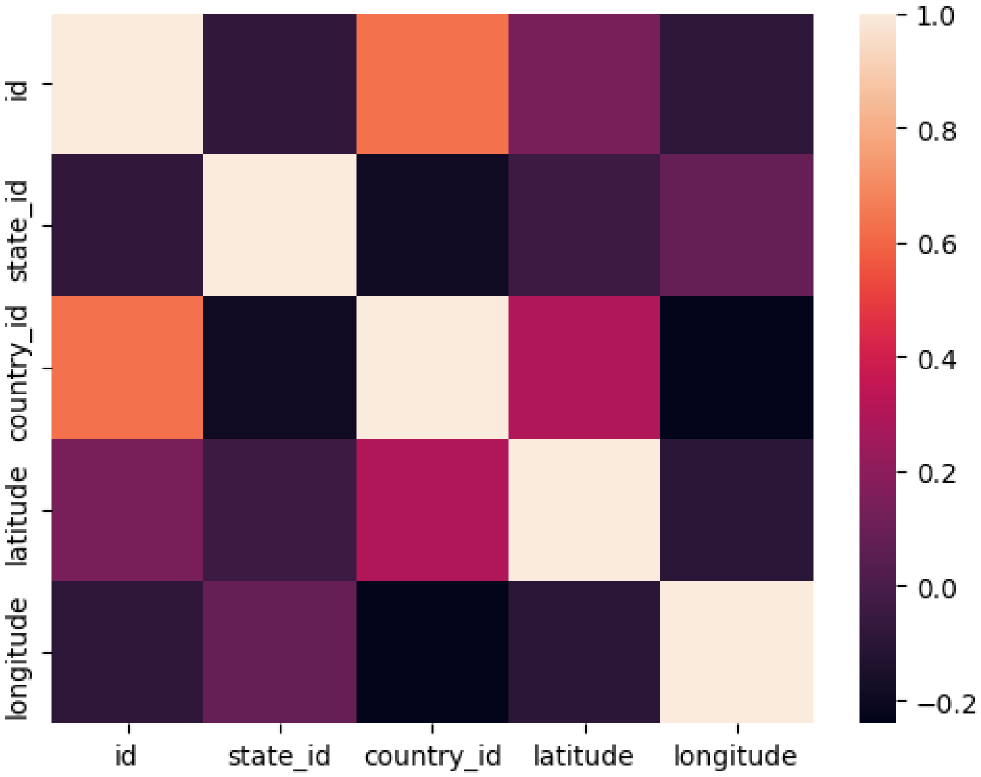
```
[21]:              id  state_id  country_id  latitude  longitude
      0            52      3901           1  36.68333   71.53333
      1            68      3901           1  37.11664   70.58002
      2            78      3901           1  36.86477   70.83421
      3            84      3901           1  36.95127   72.31800
      4           115      3901           1  37.66079   70.67346
      …             …         …           …         …          …
      150449   131496      1957         247 -19.03333   29.78333
      150450   131502      1957         247 -19.78333   29.36667
      150451   131503      1957         247 -19.67016   30.00589
      150452   131504      1957         247 -19.75000   30.16667
      150453   131508      1957         247 -20.30345   30.07514

      [150454 rows x 5 columns]
```

[22]:

```
sns.heatmap(df1.corr())
```

[22]:

```
<Axes: >
```



[23]:

```
x=df1[['id', 'state_id', 'country_id',
       'latitude']]
y=df1['longitude']
```

[24]:

```
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)
```

[25]:

```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

[25]:

```
LinearRegression()
```

[26]:

```
print(lr.intercept_)
```

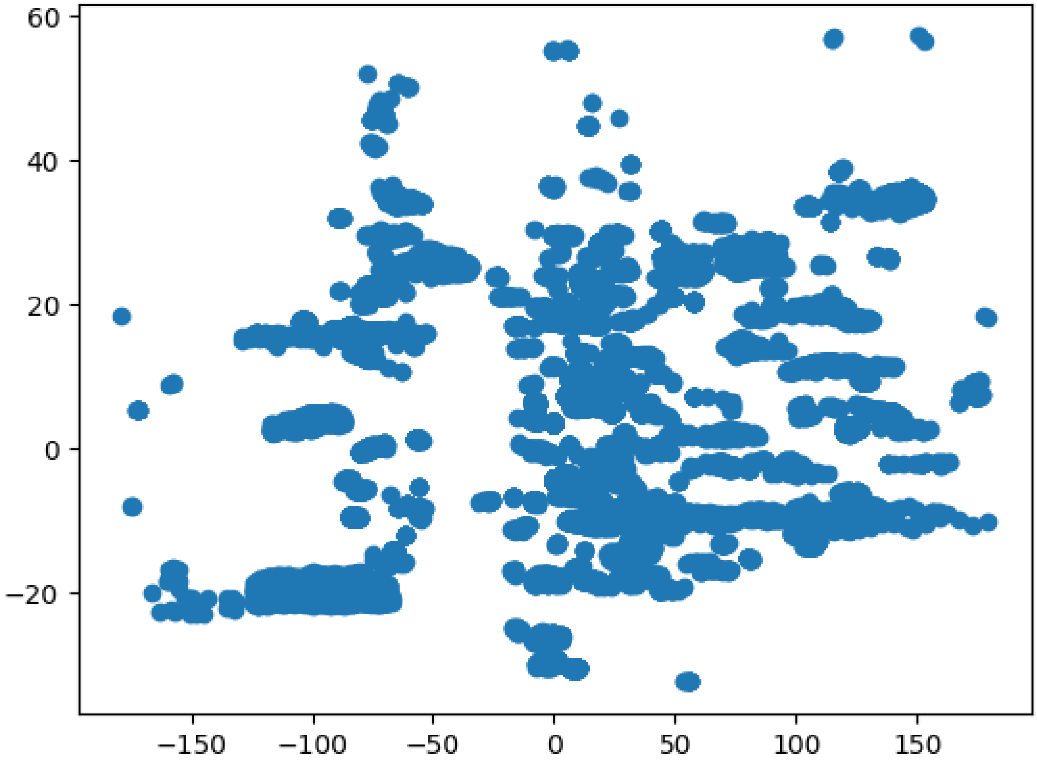
```
26.81844254985726
```

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

```
[27]:             Co-efficient
      id              0.000154
      state_id        0.002057
      country_id     -0.276313
      latitude       -0.093040
```

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

```
[28]: <matplotlib.collections.PathCollection at 0x7eea1d2451e0>
```



```
[29]: lr.score(x_test,y_test)
```

```
[29]: 0.06621717027534668
```

```
[30]: lr.score(x_train,y_train)
```

```
[30]: 0.06723235555568874
```

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

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

```
[32]: Ridge(alpha=10)
```

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

```
[33]: 0.06621717026552343
```

```
[34]: rr.score(x_train,y_train)
```

```
[34]: 0.06723235555568852
```

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

```
[35]: Lasso(alpha=10)
```

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

```
[36]: 0.06614419137530625
```

```
[37]: la.score(x_train,y_train)
```

```
[37]: 0.06718884687477733
```

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

```
[38]: ElasticNet()
```

```
[39]: en.coef_
```

```
[39]: array([ 1.53903632e-04,  2.05763581e-03, -2.76206870e-01, -9.20347347e-02])
```

```
[40]: en.intercept_
```

```
[40]: 26.782642464256952
```

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

```
[41]: array([ 25.71503816,   9.90557121,  26.88287221, …,  13.69451066,
             -29.83450702,  15.55788724])
```

[42]:
```
en.score(x_test,y_test)
```

[42]:
```
0.06621553123514157
```

[43]:
```
from sklearn import metrics
```

[44]:
```
print("Mean Absolute Error: ", metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error:  51.58112642435594
```

[45]:
```
print("Mean Squared Error: ", metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error:  4321.822112329533
```

[46]:
```
print("Root Mean Squared Error: ", np.sqrt(metrics.
  ↪mean_squared_error(y_test,prediction)))
```

```
Root Mean Squared Error:  65.74056671743509
```

[47]:
```
import pickle
filename='prediction'
pickle.dump(lr,open(filename,'wb'))
```

[49]:
```
model = pickle.load(open(filename, 'rb'))
real=[[10,20,1,20],[11,23,66,2]]
result = model.predict(real)
result
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

[49]:
```
array([24.72400315,  8.44468057])
```