8fg0kzm5z

July 31, 2023

[7]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[8]:
```
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).
```

[9]:
```
df=pd.read_csv("/content/drive/MyDrive/mydatasets/22_countries.csv")
df
```

[9]:
```
      id                       name iso3 iso2  numeric_code phone_code  \
0      1                Afghanistan  AFG   AF             4         93
1      2              Aland Islands  ALA   AX           248    +358-18
2      3                    Albania  ALB   AL             8        355
3      4                    Algeria  DZA   DZ            12        213
4      5             American Samoa  ASM   AS            16     +1-684
..   …                        …  …  …           …        …
245  243  Wallis And Futuna Islands  WLF   WF           876        681
246  244             Western Sahara  ESH   EH           732        212
247  245                      Yemen  YEM   YE           887        967
248  246                     Zambia  ZMB   ZM           894        260
249  247                   Zimbabwe  ZWE   ZW           716        263

       capital currency    currency_name currency_symbol  tld  \
0        Kabul      AFN   Afghan afghani                   .af
1    Mariehamn      EUR             Euro               €  .ax
2       Tirana      ALL     Albanian lek             Lek  .al
3      Algiers      DZD   Algerian dinar                  .dz
4    Pago Pago      USD        US Dollar               $  .as
..         …      …             …             …  …
245   Mata Utu      XPF        CFP franc                   .wf
246   El-Aaiun      MAD  Moroccan Dirham             MAD  .eh
247      Sanaa      YER      Yemeni rial                   .ye
248     Lusaka      ZMW   Zambian kwacha              ZK  .zm
```

```
249     Harare      ZWL  Zimbabwe Dollar              $  .zw

               native   region        subregion  \
0                     Asia     Southern Asia
1                Åland   Europe  Northern Europe
2            Shqipëria   Europe  Southern Europe
3                    Africa   Northern Africa
4       American Samoa  Oceania        Polynesia
..                 …      …               …
245  Wallis et Futuna  Oceania        Polynesia
246                Africa   Northern Africa
247                   Asia     Western Asia
248             Zambia   Africa   Eastern Africa
249           Zimbabwe   Africa   Eastern Africa

                                             timezones   latitude  longitude  \
0    [{zoneName:'Asia\/Kabul',gmtOffset:16200,gmtOf…  33.000000       65.0
1    [{zoneName:'Europe\/Mariehamn',gmtOffset:7200,…  60.116667       19.9
2    [{zoneName:'Europe\/Tirane',gmtOffset:3600,gmt…  41.000000       20.0
3    [{zoneName:'Africa\/Algiers',gmtOffset:3600,gm…  28.000000        3.0
4    [{zoneName:'Pacific\/Pago_Pago',gmtOffset:-396… -14.333333     -170.0
..                                                 …          …        …
245  [{zoneName:'Pacific\/Wallis',gmtOffset:43200,g… -13.300000     -176.2
246  [{zoneName:'Africa\/El_Aaiun',gmtOffset:3600,g…  24.500000      -13.0
247  [{zoneName:'Asia\/Aden',gmtOffset:10800,gmtOff…  15.000000       48.0
248  [{zoneName:'Africa\/Lusaka',gmtOffset:7200,gmt… -15.000000       30.0
249  [{zoneName:'Africa\/Harare',gmtOffset:7200,gmt… -20.000000       30.0

    emoji           emojiU
0          U+1F1E6 U+1F1EB
1          U+1F1E6 U+1F1FD
2          U+1F1E6 U+1F1F1
3          U+1F1E9 U+1F1FF
4          U+1F1E6 U+1F1F8
..    …                 …
245        U+1F1FC U+1F1EB
246        U+1F1EA U+1F1ED
247        U+1F1FE U+1F1EA
248        U+1F1FF U+1F1F2
249        U+1F1FF U+1F1FC

[250 rows x 19 columns]
```

```
[10]: df.head()
```

```
[10]:    id          name iso3 iso2  numeric_code phone_code    capital currency  \
      0   1   Afghanistan  AFG   AF             4         93      Kabul      AFN
```

```
1   2   Aland Islands  ALA   AX           248    +358-18  Mariehamn      EUR
2   3         Albania  ALB   AL             8        355     Tirana      ALL
3   4         Algeria  DZA   DZ            12        213    Algiers      DZD
4   5  American Samoa  ASM   AS            16     +1-684  Pago Pago      USD

    currency_name currency_symbol  tld          native   region  \
0  Afghan afghani                 .af                 Asia
1            Euro               €  .ax           Åland   Europe
2    Albanian lek             Lek  .al       Shqipëria   Europe
3  Algerian dinar                .dz                 Africa
4       US Dollar               $  .as  American Samoa  Oceania

         subregion                                           timezones  \
0    Southern Asia  [{zoneName:'Asia\/Kabul',gmtOffset:16200,gmtOf…
1  Northern Europe  [{zoneName:'Europe\/Mariehamn',gmtOffset:7200,…
2  Southern Europe  [{zoneName:'Europe\/Tirane',gmtOffset:3600,gmt…
3  Northern Africa  [{zoneName:'Africa\/Algiers',gmtOffset:3600,gm…
4        Polynesia  [{zoneName:'Pacific\/Pago_Pago',gmtOffset:-396…

    latitude  longitude emoji           emojiU
0  33.000000       65.0       U+1F1E6 U+1F1EB
1  60.116667       19.9       U+1F1E6 U+1F1FD
2  41.000000       20.0       U+1F1E6 U+1F1F1
3  28.000000        3.0       U+1F1E9 U+1F1FF
4 -14.333333     -170.0       U+1F1E6 U+1F1F8
```

1 Data Cleaning and Data Preprocessing

[11]:
```
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
```

[12]:

```
df.describe()
```

[12]:

```
               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
```

[13]:

```
df.columns
```

[13]:

```
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')
```

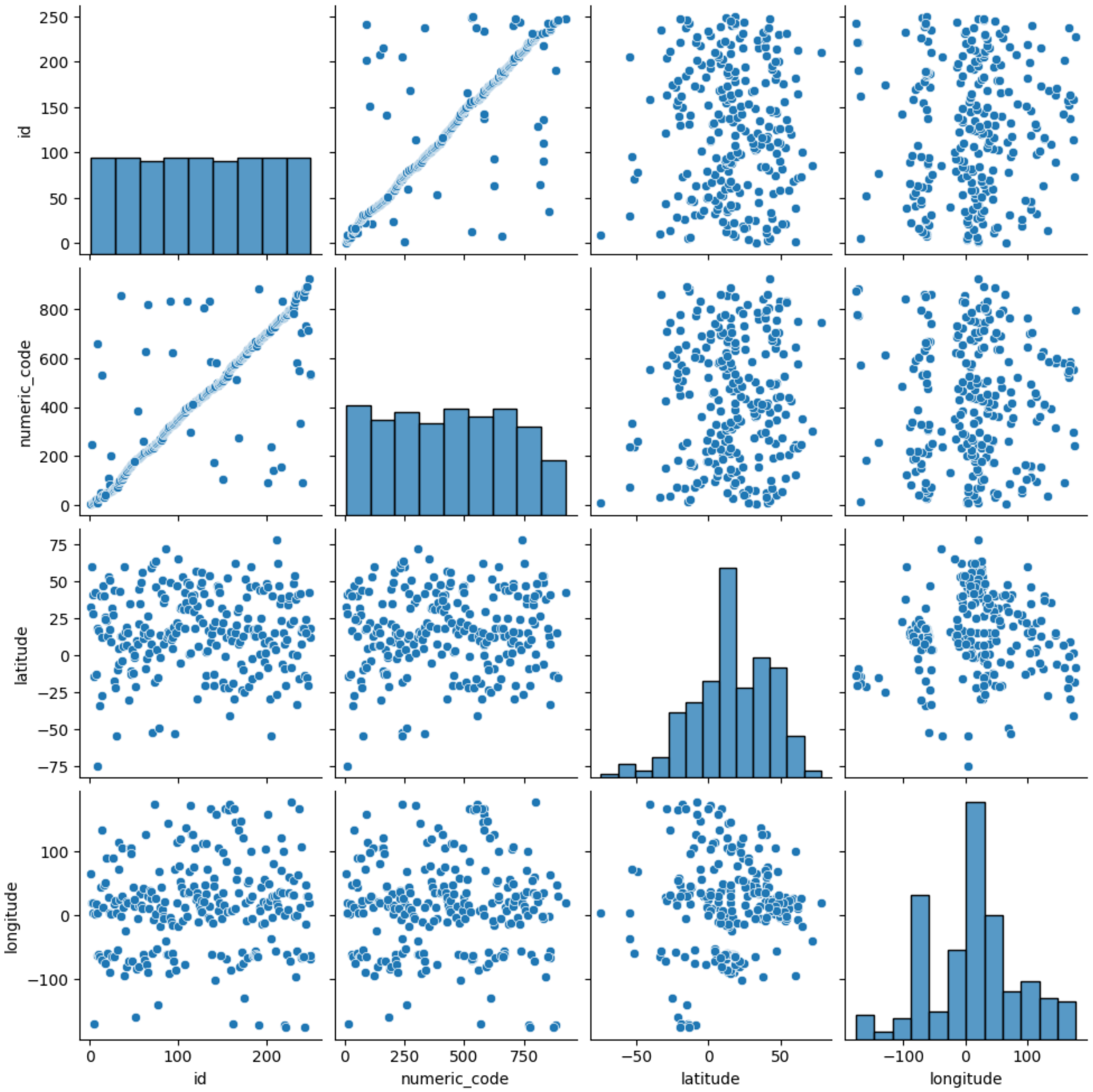
2 EDA and Visualization

[14]:

```
sns.pairplot(df)
```

[14]:

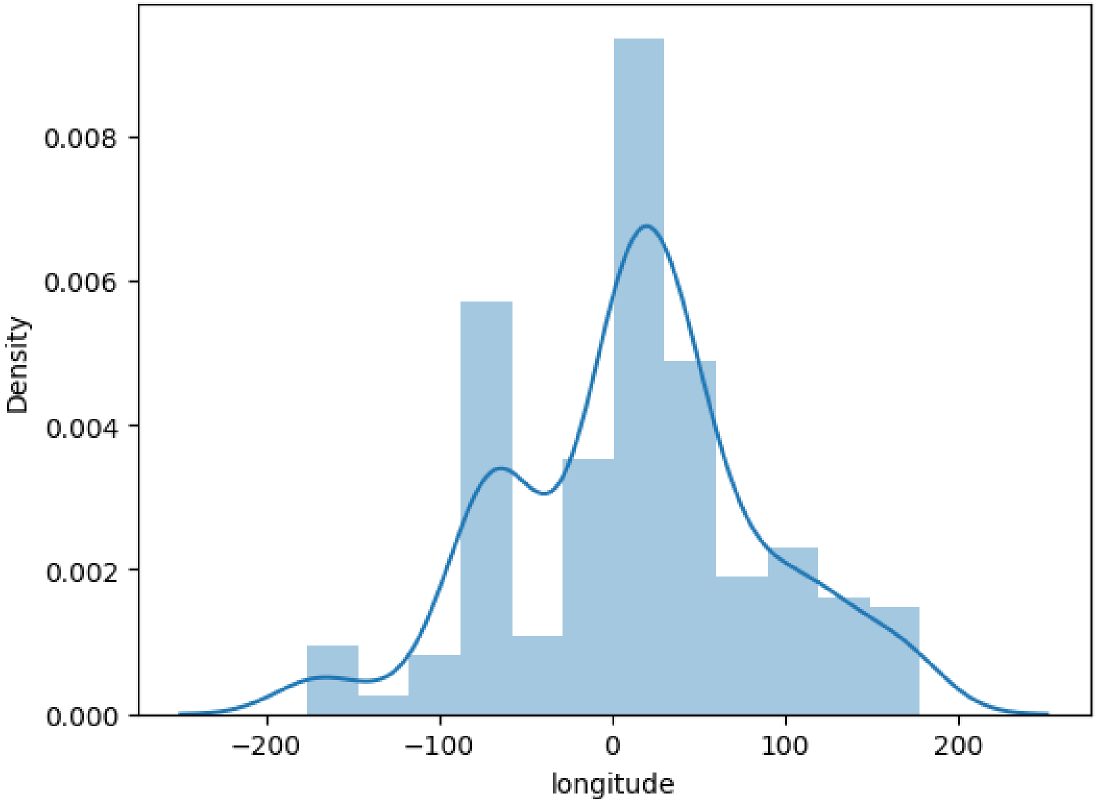
```
<seaborn.axisgrid.PairGrid at 0x7e48a8969a20>
```



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

```
<ipython-input-15-4c5c6f107715>: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'])
```

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

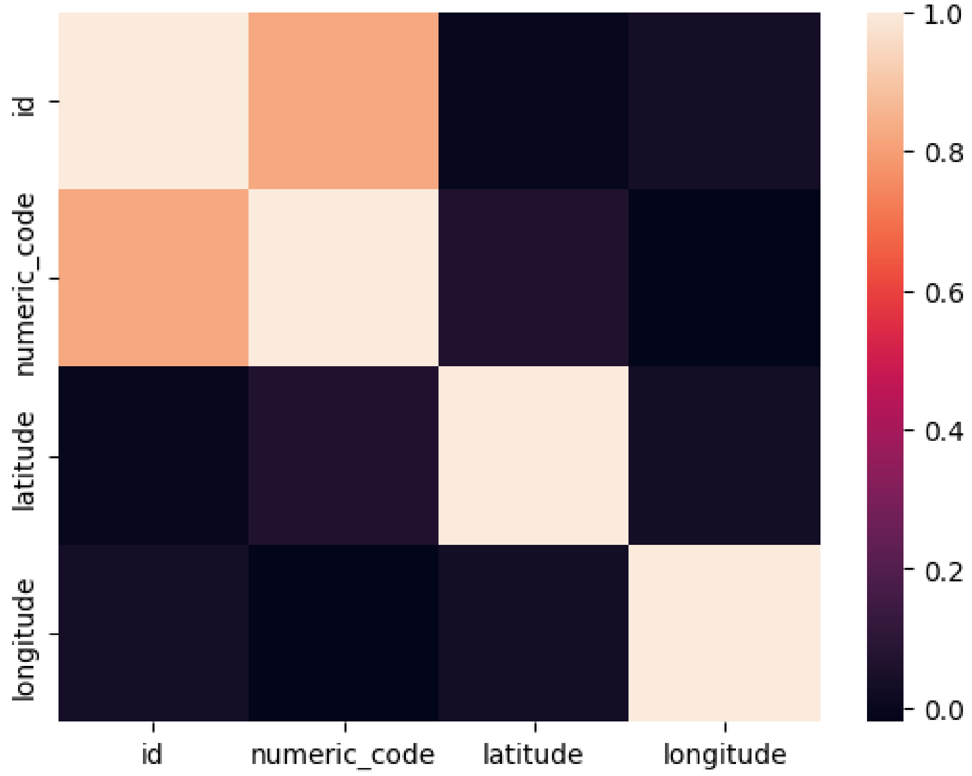


[16]:
```
df1=df[['id','numeric_code', 'latitude', 'longitude']]
df1
```

[16]:
```
      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 x 4 columns]
```

[17]:
```
sns.heatmap(df1.corr())
```

[17]: <Axes: >



[18]:
```
x=df1[['id','numeric_code', 'latitude']]
y=df1['longitude']
```

[19]:
```
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)
```

[20]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

[20]:
```
LinearRegression()
```

[21]:
```
print(lr.intercept_)
```

```
9.24756110032656
```

[22]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

[22]:

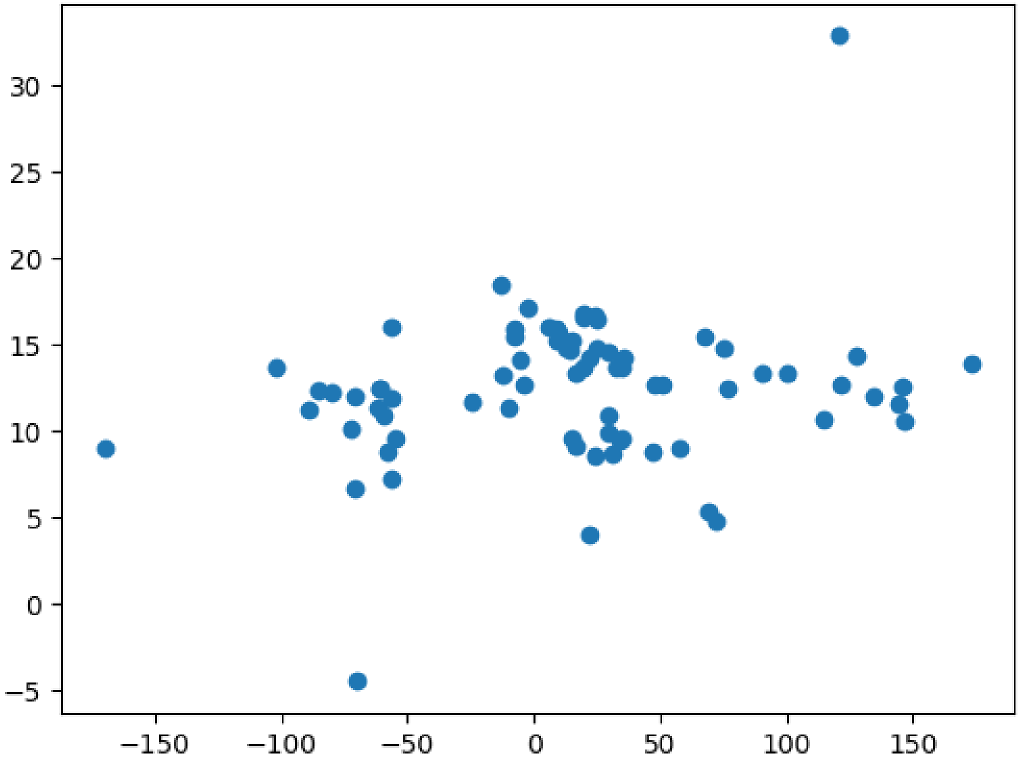
```
                Co-efficient
id                  0.120467
numeric_code       -0.031209
latitude            0.106156
```

[23]:

```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

[23]:

```
<matplotlib.collections.PathCollection at 0x7e489d28e8f0>
```



[24]:

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

[24]:

```
0.020214282786987 78
```

[25]:

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

[25]:

```
0.006070210320557434
```

[26]:

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

[27]:

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

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

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

```
[28]: 0.02021367175113542
```

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

```
[29]: 0.006070210305520685
```

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

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

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

```
[31]: 0.01910812291362196
```

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

```
[32]: 0.0060280103999195145
```

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

```
[33]: ElasticNet()
```

```
[34]: en.coef_
```

```
[34]: array([ 0.12008244, -0.03110431,  0.10530327])
```

```
[35]: en.intercept_
```

```
[35]: 9.264537380671829
```

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

```
[36]: array([15.89697488, 11.23015295, 12.67089952, 12.24363451, 12.47077855,
              6.66121143,  8.62350019, 12.6164041 ,  7.32140493, 15.15398186,
             13.3784481 , 10.93563319, 13.19420189,  9.85967827, 12.42579498,
             10.56643768, 14.7649501 ,  9.57013269, 16.00574016,  8.66545413,
             15.23311907, 13.61510864, 12.03214058,  9.5541292 , 12.65113159,
             11.39266148,  9.01010883, 32.76248961, 17.11788031, 14.54814853,
              8.83415584, 12.39541688, 14.78120231,  5.35766089, 15.38881248,
             15.93618662, 14.63000845, 14.23125461, 15.39393645, 12.05408854,
             11.88587358,  8.98416373, 13.68373316, 18.37622796, 13.69338444,
             16.6060118 , 13.6984063 , -4.4366965 ,  9.45238614, 13.89623932,
             16.71456217, 16.53945486,  4.05918149, 14.07396214, 16.36407217,
             13.29184498,  9.62678895,  4.81200829,  9.15057233,  8.77224966,
             10.16197492, 12.67476147, 10.87658947, 11.54099671, 13.2880713 ,
             10.71510886, 12.51816758, 12.37038299, 11.64691845, 14.33755451,
             15.88346382, 14.80421094, 15.62848939, 14.2470795 , 11.38986191])
```

[37]:

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

[37]:

```
0.020155445810723704
```

[38]:

```
from sklearn import metrics
```

[39]:

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

```
Mean Absolute Error:  50.43961554465261
```

[40]:

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

```
Mean Squared Error:  4380.921206445248
```

[41]:

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

```
Root Mean Squared Error:  66.18852775553516
```

[42]:

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

[43]:

```
model = pickle.load(open(filename, 'rb'))
real=[[10,20,30],[52,23,66]]
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(
```

[43]:

```
array([13.01274813, 21.80037036])
```