```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt


from sklearn.linear_model import ElasticNet
from sklearn import metrics


from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression,Ridge,Lasso


df=pd.read_csv("/content/19_nuclear_explosions.csv")
df
```

| | WEAPON SOURCE COUNTRY | WEAPON DEPLOYMENT LOCATION | Data.Source | Location.Cordinates.Latitude | Location.Cordinates.Lo |
|---|---|---|---|---|---|
| **0** | USA | Alamogordo | DOE | 32.54 | |
| **1** | USA | Hiroshima | DOE | 34.23 | |
| **2** | USA | Nagasaki | DOE | 32.45 | |
| **3** | USA | Bikini | DOE | 11.35 | |
| **4** | USA | Bikini | DOE | 11.35 | |
| **...** | ... | ... | ... | ... | |
| **2041** | CHINA | Lop Nor | HFS | 41.69 | |
| **2042** | INDIA | Pokhran | HFS | 27.07 | |
| **2043** | INDIA | Pokhran | NRD | 27.07 | |
| **2044** | PAKIST | Chagai | HFS | 28.90 | |
| **2045** | PAKIST | Kharan | HFS | 28.49 | |

2046 rows × 16 columns

```
df1=df.drop(["WEAPON SOURCE COUNTRY","WEAPON DEPLOYMENT LOCATION","Data.Source","Data.Purpose","Data.Type","Data.Name"],axis=1)
df1
```

| | Location.Cordinates.Latitude | Location.Cordinates.Longitude | Data.Magnitude.Body | Data |
|---|---|---|---|---|
| **0** | 32.54 | -105.57 | 0.0 | |
| **1** | 34.23 | 132.27 | 0.0 | |
| **2** | 32.45 | 129.52 | 0.0 | |
| **3** | 11.35 | 165.20 | 0.0 | |
| **4** | 11.35 | 165.20 | 0.0 | |
| **...** | ... | ... | ... | |
| **2041** | 41.69 | 88.35 | 5.3 | |
| **2042** | 27.07 | 71.70 | 5.3 | |
| **2043** | 27.07 | 71.70 | 0.0 | |
| **2044** | 28.90 | 64.89 | 0.0 | |
| **2045** | 28.49 | 63.78 | 5.0 | |

2046 rows × 10 columns

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2046 entries, 0 to 2045
```

```
Data columns (total 10 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   Location.Cordinates.Latitude   2046 non-null   float64
 1   Location.Cordinates.Longitude  2046 non-null   float64
 2   Data.Magnitude.Body            2046 non-null   float64
 3   Data.Magnitude.Surface         2046 non-null   float64
 4   Location.Cordinates.Depth      2046 non-null   float64
 5   Data.Yeild.Lower               2046 non-null   float64
 6   Data.Yeild.Upper               2046 non-null   float64
 7   Date.Day                       2046 non-null   int64
 8   Date.Month                     2046 non-null   int64
 9   Date.Year                      2046 non-null   int64
dtypes: float64(7), int64(3)
memory usage: 160.0 KB
```

```
y=df1["Date.Year"]
x=df1.drop(["Date.Year"],axis=1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

```
x.columns
```

```
Index(['Location.Cordinates.Latitude', 'Location.Cordinates.Longitude',
       'Data.Magnitude.Body', 'Data.Magnitude.Surface',
       'Location.Cordinates.Depth', 'Data.Yeild.Lower', 'Data.Yeild.Upper',
       'Date.Day', 'Date.Month'],
      dtype='object')
```

```
model=LinearRegression()
model.fit(x_train,y_train)
model.intercept_
```

```
1967.8970348247356
```

```
coeff=pd.DataFrame(model.coef_,x.columns,columns=["Coefficient"])
coeff
```

| | Coefficient |
|---|---|
| **Location.Cordinates.Latitude** | -0.090650 |
| **Location.Cordinates.Longitude** | -0.012253 |
| **Data.Magnitude.Body** | 2.081289 |
| **Data.Magnitude.Surface** | 1.058903 |
| **Location.Cordinates.Depth** | 0.048495 |
| **Data.Yeild.Lower** | 0.000133 |
| **Data.Yeild.Upper** | -0.000444 |
| **Date.Day** | 0.016121 |
| **Date.Month** | 0.095681 |

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

```
<matplotlib.collections.PathCollection at 0x796683794f40>
```

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

```
0.2967410711644618
```

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

```
0.2969076403223415
0.1721980630232589
```

1950

```
en=ElasticNet()
en.fit(x_train,y_train)
print(en.coef_)
print(en.intercept_)
print(en.predict(x_test))
print(en.score(x_test,y_test))
```

```
 1974.28109591 1970.42316492 1982.84843071 1964.58497387 1982.99377049
 1972.40295473 1967.07404806 1975.92654937 1977.53441299 1976.20301939
 1962.21156731 1961.67132028 1972.44104907 1963.92671211 1978.58901991
 1967.16541916 1981.42153682 1966.46399512 1974.03434901 1982.25118619
 1970.48883393 1967.17410246 1963.9629938  1976.25055183 1967.57632059
 1967.18067173 1966.39823266 1976.89943698 1982.21597533 1977.8614403
 1967.33307322 1973.33863767 1977.78693096 1986.26366797 1963.85730389
 1966.79728141 1978.92036787 1972.45736847 1967.22937583 1964.69443618
 1967.25550422 1964.8897405  1976.60124876 1967.53459659 1974.51566331
 1967.41611675 1977.12338904 1964.83594692 1976.9720927  1967.53213405
 1970.54721441 1977.80326684 1969.74688954 1972.45585347 1967.37369721
 1976.37979018 1961.27801533 1963.95686277 1974.63287276 1964.34677123
 1977.00733621 1967.14116427 1966.36938541 1977.33237457 1967.41043093
 1967.04087182 1974.89241033 1972.790526   1964.22103207 1974.40616194
 1963.89918213 1975.73690473 1978.86710233 1964.01183447 1979.36666719
 1977.61490397 1967.48881922 1975.8156761  1975.45937979 1967.25931267
 1966.34422533 1967.44108703 1963.89556574 1966.15206084 1967.49436344
 1975.47263056 1967.34860812 1978.43398949 1978.68020047 1980.23636142
 1976.54877614 1967.03725105 1976.94121076 1964.33210119 1975.1642929
 1983.19462975 1951.81936142 1981.24064388 1966.66237816 1967.32601033
 1961.6028091  1967.07336289 1975.84466363 1973.41931536 1982.58104726
 1976.59078043 1975.95304834 1967.04241761 1976.01272791 1967.11706008
 1979.35225288 1974.86891526 1978.37914899 1967.21635453 1972.35047951
 1971.78290189 1974.65946102 1979.70005369 1967.46218017 1976.87926504
 1962.61841373 1967.2906159  1974.39222767 1972.43291637 1967.46357419
 1972.32365125 1960.47704099 1976.04240601 1964.66870219 1962.22947668
 1964.12973361 1962.18338356 1967.40058185 1967.28182999 1962.56623909
 1980.74781095 1975.707301   1965.22069646 1977.85363852 1964.37057336
 1974.68655413 1969.08172567 1962.33690335 1972.50785475 1966.19916093
 1964.25098339 1967.38718773 1967.33054792 1976.87303604 1962.09170389
 1964.07702955 1974.22326237 1972.63691854 1977.32397654 1963.91095715
 1978.16057177 1976.17562223 1977.95800602 1967.17871329 1964.06222498
 1973.80292719 1975.22349181 1963.92256919 1975.66561929 1973.94067642
 1967.96122248 1972.87176946 1967.44108703 1963.96437705 1980.61700483
 1962.21597496 1966.4687959  1981.49454037 1973.24656968 1964.04450721
 1974.41161769 1973.28574945 1967.49348627 1972.23616252 1962.52709811
 1967.33177054 1975.18362527 1972.75736824 1978.3753114  1971.92278263
 1967.21838771 1975.40381474 1967.46402413 1972.7254551  1980.47630046
 1976.58442981 1972.34728604 1976.07824123 1972.26964434 1964.03286927
 1967.22793069 1967.11923709 1974.23183625 1967.30142418 1978.47286015
 1967.09283643 1972.56483777 1966.9640464  1976.41563835 1986.70173231
 1967.21432136 1967.35267447 1964.17312348 1972.76081502 1974.39809875
 1969.59983154 1964.28484326 1969.05196264 1980.4483362  1977.75865073
 1974.96090282 1964.38507777 1962.57448017 1972.6713242  1967.38170523
 1977.47058316 1972.52946163 1964.13306946 1966.93371314 1976.22355849
 1981.89186904 1967.24002213 1980.93020199 1967.41949569 1978.97943452
 1964.54334281 1964.2873607  1972.20354534 1978.33085715 1963.97262501
 1966.33729266 1964.57514077 1972.5544319  1967.23971889 1975.32309938
 1974.25138832 1978.16543509 1976.84930063 1967.13428849 1964.11463295
 1978.74325997 1964.34356974 1975.91824835 1976.32500451 1967.54227082
 1970.47736538 1977.68092759 1967.08816729 1977.94263276 1978.66627773
 1975.02505342 1972.60574374 1967.41607543 1974.59142283 1961.38472174
 1964.06977646 1977.56934253 1967.14547291 1967.13177823 1964.10809919
 1963.80437442 1966.48589858 1977.01959544 1967.34657495 1967.46886346
 1964.64891289 1967.15508342 1974.25320479 1975.98195084 1967.12460612
 1967.28719901 1967.43905386 1973.44542162 1964.25623399 1964.35124298
 1967.22620214 1967.47549268 1961.69326388 1964.2538545 ]
0.3048233794259295
```

```
print("MAE",metrics.mean_absolute_error(y_test,prediction))
print("MSE",metrics.mean_squared_error(y_test,prediction))
```

```
print("RMSE",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
MAE 6.543046531687558
MSE 70.01629337946878
RMSE 8.3675739243504
```

```
df2=pd.read_csv("/content/20_states.csv")
df2
```

| | 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 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **5072** | 1953 | Mashonaland West Province | 247 | ZW | Zimbabwe | MW | NaN | -17.485103 |
| **5073** | 1960 | Masvingo Province | 247 | ZW | Zimbabwe | MV | NaN | -20.624151 |
| **5074** | 1954 | Matabeleland North Province | 247 | ZW | Zimbabwe | MN | NaN | -18.533157 |

```
df3=df2.drop(["name","country_code","country_name","state_code","type"],axis=1)
df3
```

| | id | country_id | latitude | longitude |
|---|---|---|---|---|
| **0** | 3901 | 1 | 36.734772 | 70.811995 |
| **1** | 3871 | 1 | 35.167134 | 63.769538 |
| **2** | 3875 | 1 | 36.178903 | 68.745306 |
| **3** | 3884 | 1 | 36.755060 | 66.897537 |
| **4** | 3872 | 1 | 34.810007 | 67.821210 |
| **...** | ... | ... | ... | ... |
| **5072** | 1953 | 247 | -17.485103 | 29.788925 |
| **5073** | 1960 | 247 | -20.624151 | 31.262637 |
| **5074** | 1954 | 247 | -18.533157 | 27.549585 |
| **5075** | 1952 | 247 | -21.052337 | 29.045993 |
| **5076** | 1957 | 247 | -19.055201 | 29.603549 |

5077 rows × 4 columns

```
df3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5077 entries, 0 to 5076
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   id          5077 non-null   int64  
 1   country_id  5077 non-null   int64  
 2   latitude    5008 non-null   float64
 3   longitude   5008 non-null   float64
dtypes: float64(2), int64(2)
memory usage: 158.8 KB
```

```
df3=df3.dropna()
```

```
y=df3['latitude']
x=df3.drop(['latitude'],axis=1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

```
x.columns
```

```
Index(['id', 'country_id', 'longitude'], dtype='object')
```

```
model1=LinearRegression()
model1.fit(x_train,y_train)
model1.intercept_
```

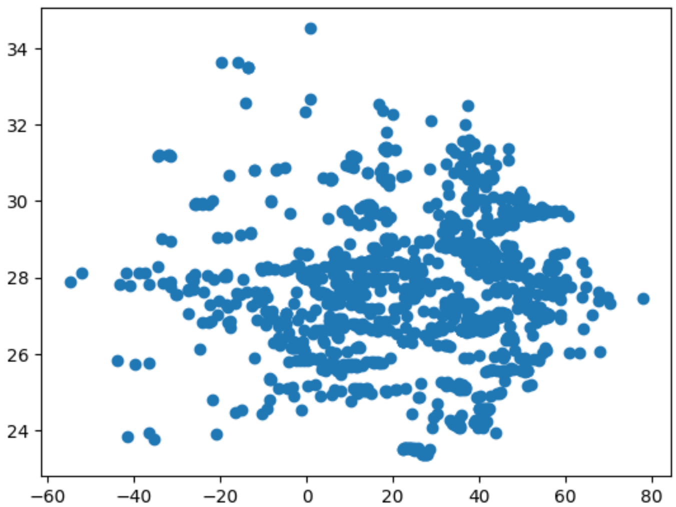
```
24.606743253805014
```

```
coeff=pd.DataFrame(model1.coef_,x.columns,columns=["Coefficient"])
coeff
```

| | Coefficient |
|---|---|
| **id** | 0.000479 |
| **country_id** | 0.016760 |
| **longitude** | -0.019788 |

```
prediction=model1.predict(x_test)
plt.scatter(y_test,prediction)
```

```
<matplotlib.collections.PathCollection at 0x79668368f610>
```




```
model1.score(x_test,y_test)
```

```
-0.0002892454306855363
```

```
rr1=Ridge(alpha=10)
rr1.fit(x_train,y_train)
la1=Lasso(alpha=10)
la1.fit(x_train,y_train)
print(rr1.score(x_test,y_test))
la1.score(x_test,y_test)
```

```
-0.00028924090645987555
0.0004674180367167935
```

```
en1=ElasticNet()
en1.fit(x_train,y_train)
print(en1.coef_)
print(en1.intercept_)
print(en1.predict(x_test))
print(en1.score(x_test,y_test))
```

```
[ 0.00047977  0.01665953 -0.01965129]
24.616119118992696
[29.98404213 28.34684722 30.70109354 ... 31.34638962 27.04996143
 25.23096051]
-0.0002466443188986478
```

```
print("MAE",metrics.mean_absolute_error(y_test,prediction))
print("MSE",metrics.mean_squared_error(y_test,prediction))
print("RMSE",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
MAE 19.10927478504828
MSE 510.9594477046008
RMSE 22.604412129153033
```

```
df4=pd.read_csv("/content/21_cities.csv")
df4
```

| | id | name | state_id | state_code | state_name | country_id | country_code | count |
|---|---|---|---|---|---|---|---|---|
| **0** | 52 | Ashkāsham | 3901 | BDS | Badakhshan | 1 | AF | Afg |
| **1** | 68 | Fayzabad | 3901 | BDS | Badakhshan | 1 | AF | Afg |
| **2** | 78 | Jurm | 3901 | BDS | Badakhshan | 1 | AF | Afg |
| **3** | 84 | Khandūd | 3901 | BDS | Badakhshan | 1 | AF | Afg |
| **4** | 115 | Rāghistān | 3901 | BDS | Badakhshan | 1 | AF | Afg |
| **...** | ... | ... | ... | ... | ... | ... | ... | |
| **150449** | 131496 | Redcliff | 1957 | MI | Midlands Province | 247 | ZW | Zi |
| **150450** | 131502 | Shangani | 1957 | MI | Midlands Province | 247 | ZW | Zi |
| **150451** | 131503 | Shurugwi | 1957 | MI | Midlands Province | 247 | ZW | Zi |
| **150452** | 131504 | Shurugwi District | 1957 | MI | Midlands Province | 247 | ZW | Zi |

```
df5=df4.drop(["name","country_name","state_code","state_name","country_code","wikiDataId"],axis=1)
df5
```

| | 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 × 5 columns

```
df5.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150454 entries, 0 to 150453
Data columns (total 5 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   id          150454 non-null  int64
 1   state_id    150454 non-null  int64
 2   country_id  150454 non-null  int64
 3   latitude    150454 non-null  float64
 4   longitude   150454 non-null  float64
dtypes: float64(2), int64(3)
memory usage: 5.7 MB
```

```
y=df5['latitude']
x=df5.drop(['latitude'],axis=1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

```
x.columns
```

```
Index(['id', 'state_id', 'country_id', 'longitude'], dtype='object')
```

```
model2=LinearRegression()
model2.fit(x_train,y_train)
model2.intercept_
```

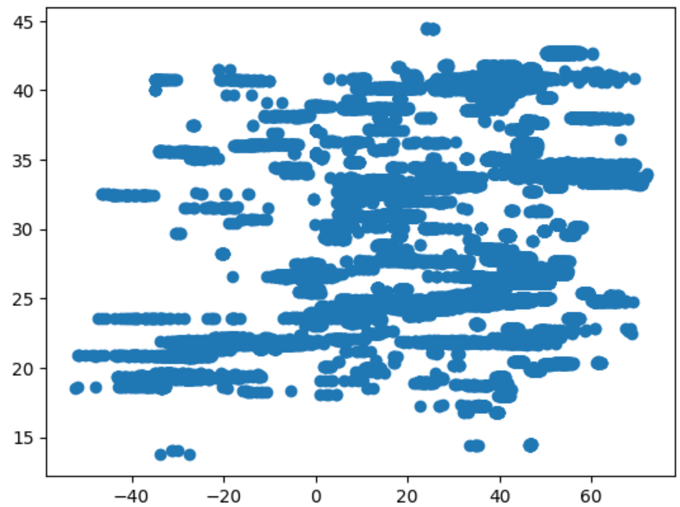
```
17.576312551809135
```

```
coeff=pd.DataFrame(model2.coef_,x.columns,columns=["Coefficient"])
coeff
```

| | Coefficient |
|---|---|
| **id** | -0.000039 |
| **state_id** | 0.000420 |
| **country_id** | 0.112229 |
| **longitude** | -0.008506 |

```
prediction=model2.predict(x_test)
plt.scatter(y_test,prediction)
```

```
<matplotlib.collections.PathCollection at 0x7966837f5900>
```




```
model2.score(x_test,y_test)
```

```
0.08734648353424512
```

```
rr2=Ridge(alpha=10)
rr2.fit(x_train,y_train)
la2=Lasso(alpha=10)
la2.fit(x_train,y_train)
print(rr2.score(x_test,y_test))
la2.score(x_test,y_test)
```

```
0.08734648381792465
0.08729261603523175
```

```
en2=ElasticNet()
en2.fit(x_train,y_train)
print(en2.coef_)
print(en2.intercept_)
print(en2.predict(x_test))
print(en2.score(x_test,y_test))
```

```
[-3.83795621e-05  4.17935861e-04  1.12059024e-01 -8.42577299e-03]
17.591497915536813
[21.88354556 31.94783655 27.50389621 ... 26.33371072 40.47686044
 22.77542612]
0.08734956605654653
```

```
print("MAE",metrics.mean_absolute_error(y_test,prediction))
print("MSE",metrics.mean_squared_error(y_test,prediction))
print("RMSE",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
MAE 17.453435631341513
MSE 474.661041843344
```

```
RMSE 21.786717096509605
```

```
df6=pd.read_csv("/content/22_countries.csv")
df6
```

| | id | name | iso3 | iso2 | numeric_code | phone_code | capital | currency | currency_nam |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | Afghanistan | AFG | AF | 4 | 93 | Kabul | AFN | Afghan afghar |
| 1 | 2 | Aland Islands | ALA | AX | 248 | +358-18 | Mariehamn | EUR | Eur |
| 2 | 3 | Albania | ALB | AL | 8 | 355 | Tirana | ALL | Albanian le |
| 3 | 4 | Algeria | DZA | DZ | 12 | 213 | Algiers | DZD | Algerian dina |
| 4 | 5 | American Samoa | ASM | AS | 16 | +1-684 | Pago Pago | USD | US Dolla |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | . |
| 245 | 243 | Wallis And Futuna Islands | WLF | WF | 876 | 681 | Mata Utu | XPF | CFP fran |
| 246 | 244 | Western Sahara | ESH | EH | 732 | 212 | El-Aaiun | MAD | Moroccai Dirhar |
| 247 | 245 | Yemen | YEM | YE | 887 | 967 | Sanaa | YER | Yemeni ria |
| 248 | 246 | Zambia | ZMB | ZM | 894 | 260 | Lusaka | ZMW | Zambiai kwach |
| 249 | 247 | Zimbabwe | ZWE | ZW | 716 | 263 | Harare | ZWL | Zimbabw Dolla |

250 rows × 19 columns

```
df7=df6.drop(["name","iso3","iso2","capital","currency_name","currency_symbol","tld","region","subregion","timezones","emoji","emojiU","currency","na
df7
```

| | 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 × 4 columns

```
df6.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
```

```
y=df7['latitude']
x=df7.drop(['latitude'],axis=1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

```
x.columns
```

```
Index(['id', 'numeric_code', 'longitude'], dtype='object')
```

```
model3=LinearRegression()
model3.fit(x_train,y_train)
model3.intercept_
```

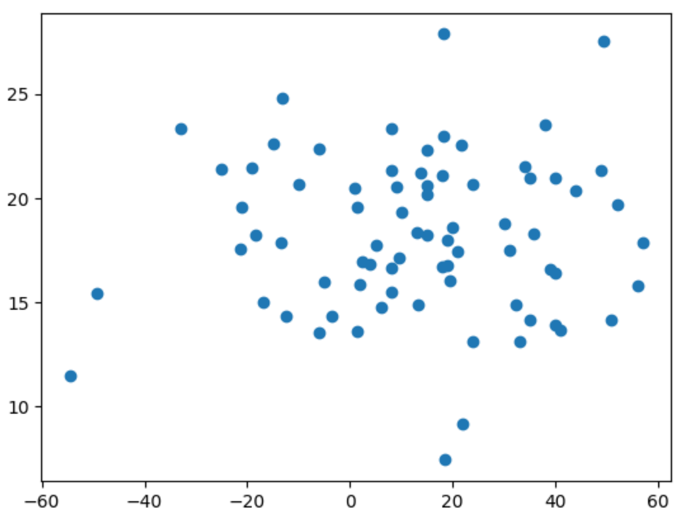
```
13.830967695856955
```

```
coeff=pd.DataFrame(model3.coef_,x.columns,columns=["Coefficient"])
coeff
```

| | Coefficient |
|---|---|
| **id** | -0.037692 |
| **numeric_code** | 0.020606 |
| **longitude** | -0.011784 |

```
prediction=model3.predict(x_test)
plt.scatter(y_test,prediction)
```

```
<matplotlib.collections.PathCollection at 0x7966829f1ed0>
```




```
model3.score(x_test,y_test)
```

```
-0.06447730154217335
```

```
rr3=Ridge(alpha=10)
rr3.fit(x_train,y_train)
la3=Lasso(alpha=10)
la3.fit(x_train,y_train)
print(rr3.score(x_test,y_test))
la3.score(x_test,y_test)
```

```
-0.06447786628956509
-0.06485532649823367
```

```
en3=ElasticNet()
en3.fit(x_train,y_train)
print(en3.coef_)
print(en3.intercept_)
print(en3.predict(x_test))
print(en3.score(x_test,y_test))
```

```
[-0.03732614  0.02051141 -0.01171587]
13.825296715904178
[13.08724191 23.35843051 17.08743602 13.64309211 15.43647209 17.73561783
 16.97443541 16.81934892 13.89625059 14.34137487 27.50386464 17.83214571
 21.52178796 21.46010076 16.41352926 14.97379835 16.54824801 16.02760418
 14.34531154 16.65794432 20.37518862  9.17407995 20.5393648  19.58029567
 27.80427027 21.37043888 13.61464743 20.15000047 21.35654857 22.38465697
 15.9962326  14.10589696 13.5571564  20.99320604 17.84941191 13.1084845
 11.50915117 15.84286479 22.51800486 17.46319352 20.9436435  19.65660591
 22.62878884 23.49432842 20.62547366 18.1919628  18.79140531 22.98777918
 16.72843898 18.23611539 17.46488696 21.35263869 15.50539898 24.78737481
 14.84285205 20.64717119 14.72768186 21.08162362 15.77226858 17.93752528
 16.72708797 20.44607517 20.65124505  7.47386094 22.31164941 19.54263887
 19.33151809 14.11523344 18.25577307 17.53024239 18.37214794 14.8814304
 21.17546443 18.58200671 23.30853073]
-0.06449532300036953
```

```
print("MAE",metrics.mean_absolute_error(y_test,prediction))
print("MSE",metrics.mean_squared_error(y_test,prediction))
print("RMSE",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
MAE 19.274448382030435
MSE 590.158391920643
RMSE 24.293175830274702
```

```
df8=pd.read_csv("/content/23_Vande Bharat.csv")
df8
```

| | Sr. No. | Train Name | Train Number | Originating City | Originating Station | Terminal City | Term |
|---|---|---|---|---|---|---|---|
| **0** | 1 | New Delhi - Varanasi Vande Bharat Express | 22435/22436 | Delhi | New Delhi | Varanasi | Va |
| **1** | 2 | New Delhi - Shri Mata Vaishno Devi Katra Vande... | 22439/22440 | Delhi | New Delhi | Katra | Sh |
| **2** | 3 | Mumbai Central - Gandhinagar Capital Vande Bha... | 20901/20902 | Mumbai | Mumbai Central | Gandhinagar | Gand |
| **3** | 4 | New Delhi - Amb Andaura Vande Bharat Express | 22447/22448 | Delhi | New Delhi | Andaura | |
| **4** | 5 | MGR Chennai Central - Mysuru Vande Bharat Express | 20607/20608 | Chennai | Chennai Central | Mysuru | M |
| **5** | 6 | Bilaspur - Nagpur Vande Bharat Express | 20825/20826 | Bilaspur, Chhattisgarh | Bilaspur Junction | Nagpur | N |
| **6** | 7 | Howrah - New Jalpaiguri Vande Bharat Express | 22301/22302 | Kolkata | Howrah Junction | Siliguri | |
| **7** | 8 | Visakhapatnam - Secunderabad Vande Bharat Express | 20833/20834 | Visakhapatnam | Visakhapatnam Junction | Hyderabad | |
| **8** | 9 | Mumbai CSMT - Solapur Vande Bharat Express | 22225/22226 | Mumbai | Chhatrapati Shivaji Terminus | Solapur | |
| **9** | 10 | Mumbai CSMT - Sainagar Shirdi Vande Bharat Exp... | 22223/22224 | Mumbai | Chhatrapati Shivaji Terminus | Shirdi | |
| **10** | 11 | Rani Kamalapati (Habibganj) - Hazrat Nizamuddi... | 20171/20172 | Bhopal | Habibganj (Rani Kamalapati) | Delhi | Haz |
| **11** | 12 | Secunderabad - Tirupati Vande Bharat Express | 20701/20702 | Hyderabad | Secunderabad Junction | Tirupati | |
| **12** | 13 | MGR Chennai Central - Coimbatore Vande Bharat ... | 20643/20644 | Chennai | Chennai Central | Coimbatore | Coim |
| **13** | 14 | Delhi Cantonment - Ajmer Vande Bharat Express | 20977/20978 | Delhi | Delhi Cantonment | Ajmer | |
| **14** | 15 | Kasaragod - Thiruvananthapuram Vande Bharat Ex... | 20633/20634 | Kasaragod | Kasaragod | Thiruvananthapuram | Thiru |
| **15** | 16 | Howrah - Puri Vande | 22895/22896 | Kolkata | Howrah | Puri | |

```
df8.columns
```

```
Index(['Sr. No.', 'Train Name', 'Train Number', 'Originating City',
       'Originating Station', 'Terminal City', 'Terminal Station', 'Operator',
       'No. of Cars', 'Frequency', 'Distance', 'Travel Time', 'Speed',
       'Average Speed', 'Inauguration', 'Average occupancy'],
      dtype='object')
```

```
df9=df8.loc[:,["Sr. No.",'No. of Cars',"Average occupancy"]]
df9
```

| | Sr. No. | No. of Cars | Average occupancy |
|---|---|---|---|
| 0 | 1 | 16 | 126% |
| 1 | 2 | 16 | 114% |
| 2 | 3 | 16 | 132% |
| 3 | 4 | 16 | 70% |
| 4 | 5 | 16 | 75% |
| 5 | 6 | 8 | 96% |
| 6 | 7 | 16 | 100% |
| 7 | 8 | 16 | 120% |
| 8 | 9 | 16 | 100% |
| 9 | 10 | 16 | 93% |
| 10 | 11 | 16 | 90% |
| 11 | 12 | 16 | 110% |
| 12 | 13 | 8 | 150% |
| 13 | 14 | 16 | 70% |
| 14 | 15 | 16 | 177% |
| 15 | 16 | 16 | 99% |
| 16 | 17 | 8 | 100% |
| 17 | 18 | 8 | 91% |
| 18 | 19 | 16 | 94% |
| 19 | 19 | 16 | 94% |
| 20 | 20 | 8 | 118% |

```
df9=df9.replace("%","",regex=True).astype(int)
df9
```

| | Sr. No. | No. of Cars | Average occupancy |
|---|---|---|---|
| **0** | 1 | 16 | 126 |
| **1** | 2 | 16 | 114 |

```
df9.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26 entries, 0 to 25
Data columns (total 3 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Sr. No.            26 non-null     int64
 1   No. of Cars        26 non-null     int64
 2   Average occupancy  26 non-null     int64
dtypes: int64(3)
memory usage: 752.0 bytes
```

| | | | |
|---|---|---|---|
| **9** | 10 | 16 | 93 |

```
y=df9["No. of Cars"]
x=df9.drop(["No. of Cars"],axis=1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

```
x.columns
```

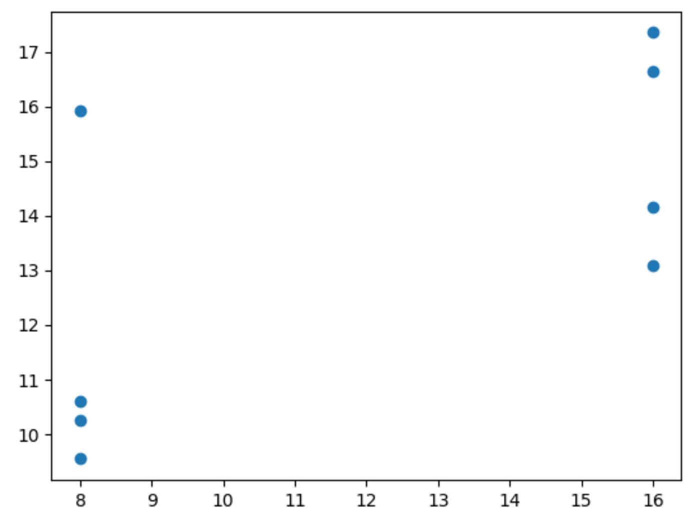
```
Index(['Sr. No.', 'Average occupancy'], dtype='object')
```

```
model4=LinearRegression()
model4.fit(x_train,y_train)
model4.intercept_
```

```
18.028982791702994
```

```
prediction=model4.predict(x_test)
plt.scatter(y_test,prediction)
```

```
<matplotlib.collections.PathCollection at 0x796682a7d090>
```




```
model4.score(x_test,y_test)
```

```
0.28753033682015605
```

```
rr4=Ridge(alpha=10)
rr4.fit(x_train,y_train)
la4=Lasso(alpha=10)
la4.fit(x_train,y_train)
print(rr4.score(x_test,y_test))
la4.score(x_test,y_test)
```

```
0.2907101089706162
0.1966987533496083
```

```
en4=ElasticNet()
en4.fit(x_train,y_train)
print(en4.coef_)
print(en4.intercept_)
```

```
print(en4.predict(x_test))
print(en4.score(x_test,y_test))
```

```
[-0.3382352   0.00101033]
17.73688339558089
[13.07231343 10.34016341 16.45466538  9.67278595 17.17559009 15.80446345
 10.70668772 14.10722552]
0.2948979508593491
```

```
print("MAE",metrics.mean_absolute_error(y_test,prediction))
print("MSE",metrics.mean_squared_error(y_test,prediction))
print("RMSE",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
MAE 2.6327971141764364
MSE 11.399514610877503
RMSE 3.3763167225361874
```

```
import pickle
```

```
pickle.dump(model,open("Prediction","wb"))
pickle.dump(model1,open("Prediction1","wb"))
pickle.dump(model2,open("Prediction2","wb"))
pickle.dump(model3,open("Prediction3","wb"))
pickle.dump(model4,open("Prediction4","wb"))
```

```
mod=pickle.load(open("Prediction","rb"))
mod1=pickle.load(open("Prediction1","rb"))
mod2=pickle.load(open("Prediction2","rb"))
mod3=pickle.load(open("Prediction3","rb"))
mod4=pickle.load(open("Prediction4","rb"))
```

```
val=[[3,4334,75,84,12,112,121,22,24],[7,64,84,78,45,67,54,567,67]]
mod.predict(val)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegression was fitted
  warnings.warn(
array([2162.761056  , 2241.61928827])
```

```
val1=[[3,4334,75],[7,64,84]]
mod1.predict(val1)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegression was fitted
  warnings.warn(
array([95.76218046, 24.02054049])
```

```
val2=[[3,4334,75,76],[7,64,84,4]]
mod2.predict(val2)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegression was fitted
  warnings.warn(
array([27.16628385, 26.99609518])
```

```
val3=[[3,75,76],[7,84,4]]
mod3.predict(val3)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegression was fitted
  warnings.warn(
array([14.36775787, 15.25088119])
```

```
val4=[[3,76],[7,4]]
mod4.predict(val4)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegression was fitted
  warnings.warn(
array([16.9862886, 15.5502371])
```

✓ 0s completed at 9:58 AM