```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from numpy import mean
from numpy import absolute
from numpy import sqrt
from sklearn.svm import SVR
from sklearn import linear_model
from mpl_toolkits.mplot3d import Axes3D
data=pd.read_csv('data-project-yo.csv')
data
```

```
         Country/Region  Confirmed  Deaths  Recovered/  Active  New 
cases  \
0           Afghanistan      36263    1269       25198    9796   
106   
1               Albania       4880     144        2745    1991   
117   
2               Algeria      27973    1163       18837    7973   
616   
3               Andorra        907      52         803      52   
10   
4                Angola        950      41         242     667   
18   
..                  ...        ...     ...         ...     ...   
...   
182  West Bank and Gaza      10621      78        3752    6791   
152   
183      Western Sahara         10       1           8       1   
0   
184               Yemen       1691     483         833     375   
10   
185              Zambia       4552     140        2815    1597   
71   
186            Zimbabwe       2704      36         542    2126   
192   

     New deaths  New recovered  Deaths / 100 Cases  Recovered / 100 
Cases  \
0            10             18                3.50                
69.49   
1             6             63                2.95                
56.25   
2             8            749                4.16                
67.34   
3             0              0                5.73                
88.53   
4             1              0                4.32                
```

```
25.47
..          ...            ...                  ...
...
182           2              0                 0.73
35.33
183           0              0                10.00
80.00
184           4             36                28.56
49.26
185           1            465                 3.08
61.84
186           2             24                 1.33
20.04

     Deaths / 100 Recovered  Recovery rate  Incidence rate
0                      5.04      69.486805        0.292309
1                      5.25      56.250000        2.397541
2                      6.17      67.339935        2.202123
3                      6.48      88.533627        1.102536
4                     16.94      25.473684        1.894737
..                      ...            ...             ...
182                    2.08      35.326240        1.431127
183                   12.50      80.000000        0.000000
184                   57.98      49.260792        0.591366
185                    4.97      61.840949        1.559754
186                    6.64      20.044379        7.100592

[187 rows x 13 columns]
```

Here we covered the part of comfirmed and recovered cases and a comparison between the recovery rate estimated and the actual data

```python
data2=pd.DataFrame(data[['Country/Region','Recovered/']])
plt.figure();
data2.plot.bar(title='Fig.no.1:bar chart confirmed recovery for each country',color='red');
plt.xlabel('countries');
plt.ylabel('number of recovered')
```

```
Text(0, 0.5, 'number of recovered')
```

```
<Figure size 432x288 with 0 Axes>
```

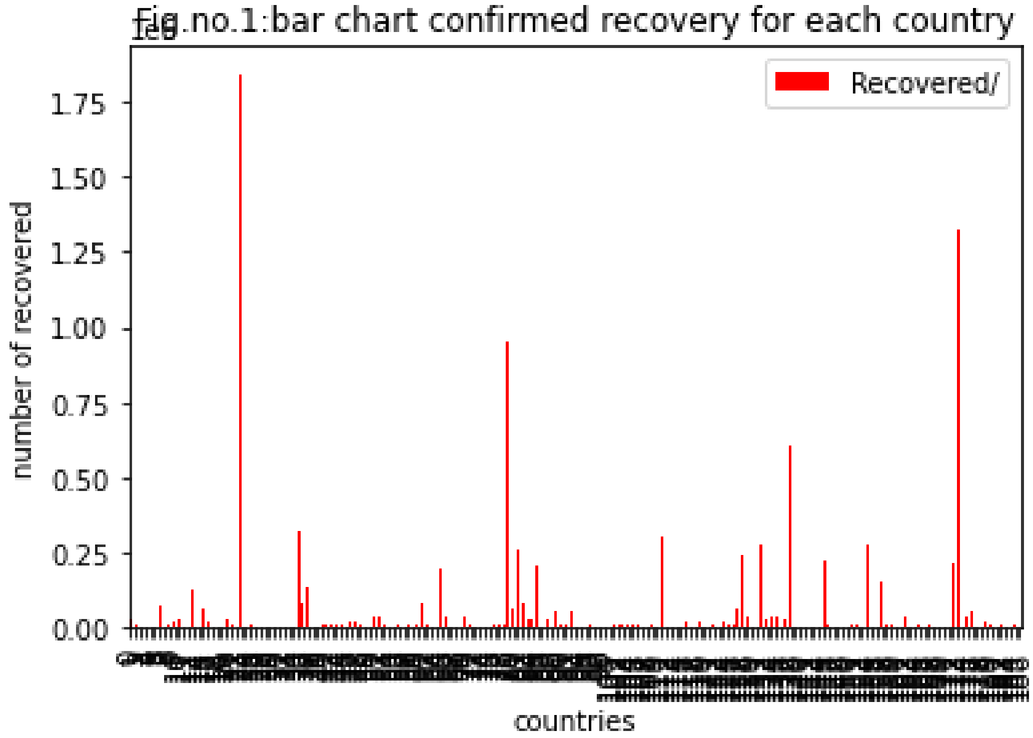
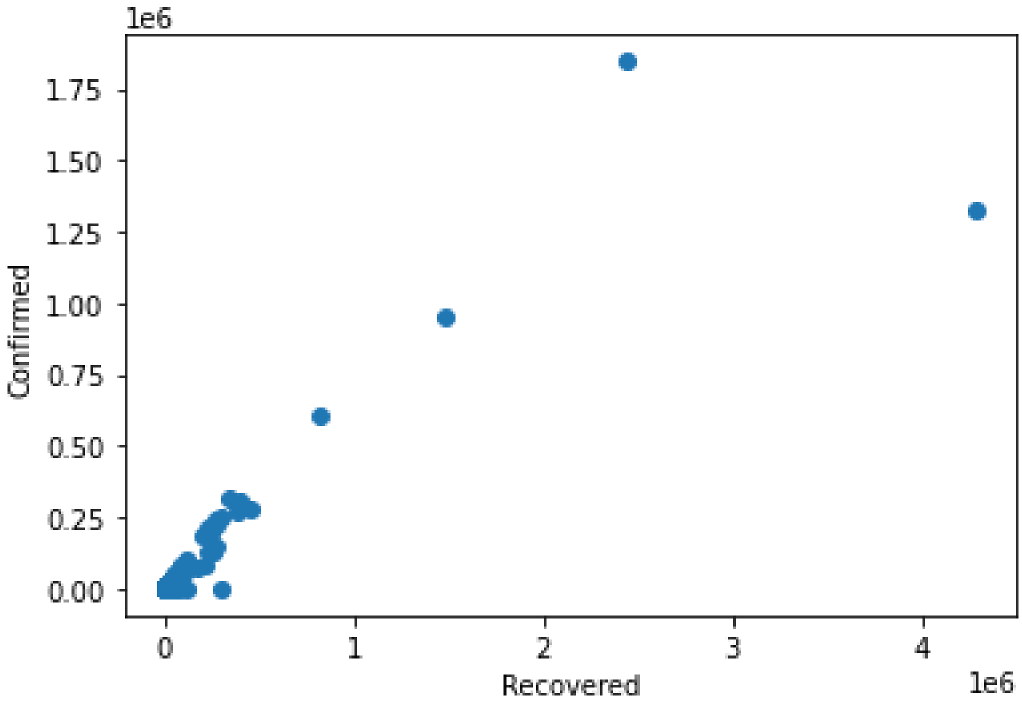


Fig.no.1:bar chart confirmed recovery for each country

here we have a graph that represents number of recovered cases for each country

```
x = data['Confirmed']
y = data['Recovered/']
plt.xlabel('Recovered')
plt.ylabel("Confirmed")
plt.scatter(x, y);
```

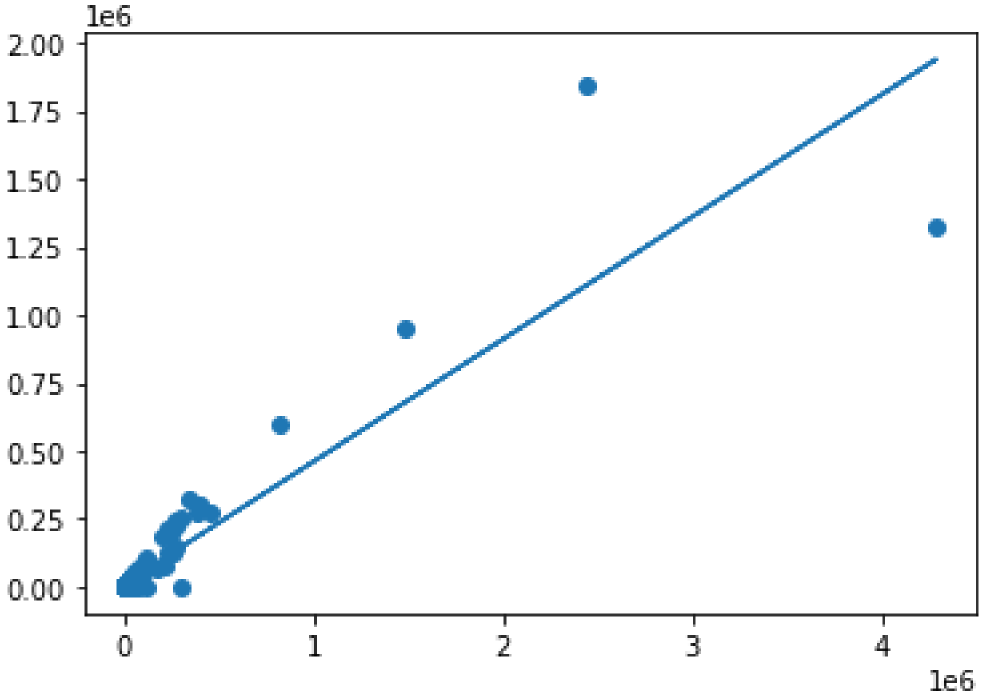


this graph shows the confirmed cases versus the recovered cases

```
from sklearn.linear_model import LinearRegression
model = LinearRegression(fit_intercept=True)
X = x[:, np.newaxis]
print(X.shape)
model.fit(X, y)
slope=model.coef_
b=model.intercept_
xfit = np.linspace(-1, 4290259)
Xfit = xfit[:, np.newaxis]
yfit = model.predict(Xfit)
plt.scatter(X, y)
plt.plot(X,b+slope*X);
```

```
(187, 1)
```

```
<ipython-input-4-c49d6c0c88fe>:3: FutureWarning: Support for multi-
dimensional indexing (e.g. `obj[:, None]`) is deprecated and will be
removed in a future version.  Convert to a numpy array before indexing
instead.
  X = x[:, np.newaxis]
```

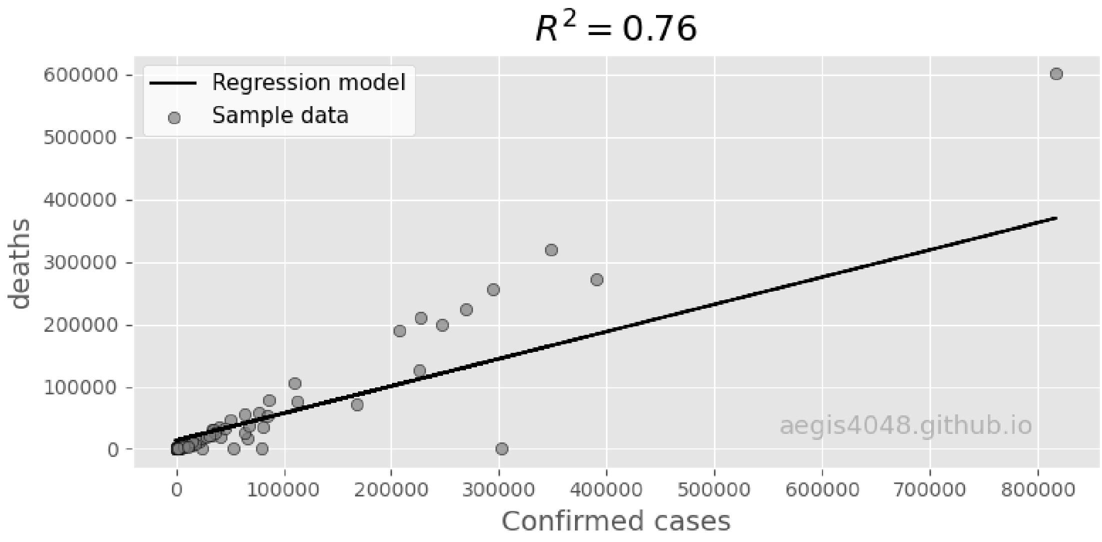


this graph represents the linear regression with the model. compared to the previous one, this graph has scattered dots as well as a plotted line .

```
from sklearn.model_selection import train_test_split
# split the data with 60% in each set
X1, X2, y1, y2 = train_test_split(x,y, random_state=1,train_size=0.4)
# we tested
from sklearn.linear_model import LinearRegression
model = LinearRegression(fit_intercept=True)
model.fit((X1.values).reshape(-1,1),(y1.values))
slope=model.coef_
b=model.intercept_
y2_model = model.predict((X2.values).reshape(-1,1))
from sklearn.metrics import r2_score
r2=r2_score((y2.values).reshape(-1,1),y2_model)
r2
```

```
0.7632948460012838
```

```
plt.style.use('default')
plt.style.use('ggplot')

fig, ax = plt.subplots(figsize=(8, 4))

ax.plot(X2, y2_model, color='k', label='Regression model')
ax.scatter(X2, y2, edgecolor='k', facecolor='grey', alpha=0.7,
label='Sample data')
ax.set_ylabel('deaths', fontsize=14)
ax.set_xlabel('Confirmed cases', fontsize=14)
ax.text(0.8, 0.1, 'aegis4048.github.io', fontsize=13, ha='center',
va='center',
         transform=ax.transAxes, color='grey', alpha=0.5)
ax.legend(facecolor='white', fontsize=11)
ax.set_title('$R^2= %.2f$' % r2, fontsize=18)

fig.tight_layout()
```



we tried different percentage of splitting: 90%-train-10%-test/ 80%-20%/ 70%-30%/ 20%-80% it gave us 0.62 accuracy 60%-40%/ 50%-50%/ 40%-60%/ 30%-70% gave us around 0.75 accuracy with 60%-40% being the highest one 0.7632948460012838 thus we conclude that the accuracy of our prediction is average, not the greatest, that is why we will try the percentage of error that cross validation is going to give us.

```
#define cross-validation method to use
cv = KFold(n_splits=10, random_state=1, shuffle=True)
#use k-fold CV to evaluate model
scores = cross_val_score(model, X, y,
scoring='neg_mean_absolute_error',
                         cv=cv, n_jobs=-1)

#view mean absolute error
sqrt(mean(absolute(scores)))
```

```
166.0520657802262
```

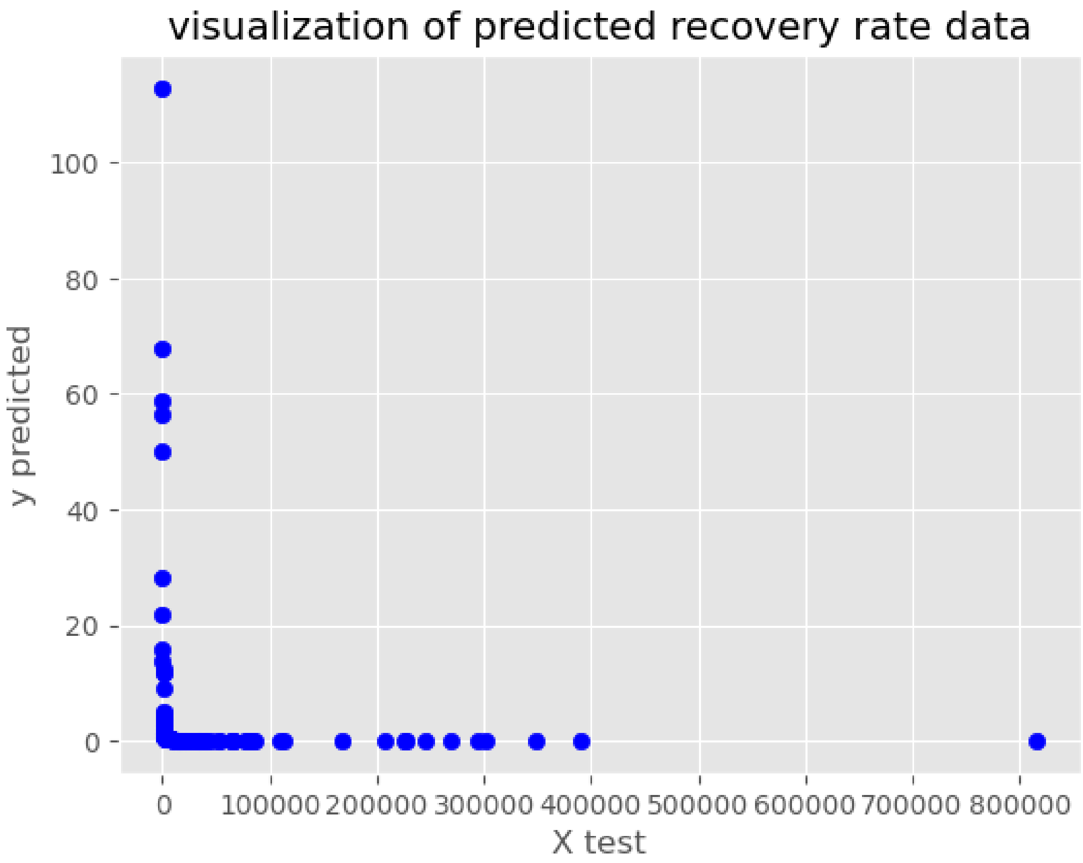
the sqrt of the MAE (mean-absolute-error) should be the lowest possible considering that we have quite a bit of data, the MAE was above 100 meaning it is not the best, also confirming what we found with the accuracy earlier being 76%.

```
rec_rate_model=y2_model/(10*X2)
rec_rate_train=y2*100/X2
```

here we wanted to show that since we have linear regression between confirmed and recovered cases that the rate of recovery between the train model and the test value will also be similar, which the following scattered plots show.

```
plt.scatter(X2, rec_rate_model, c='b', label='Train data')
plt.xlabel('X test')
plt.ylabel('y predicted')
plt.title('visualization of predicted recovery rate data')
```

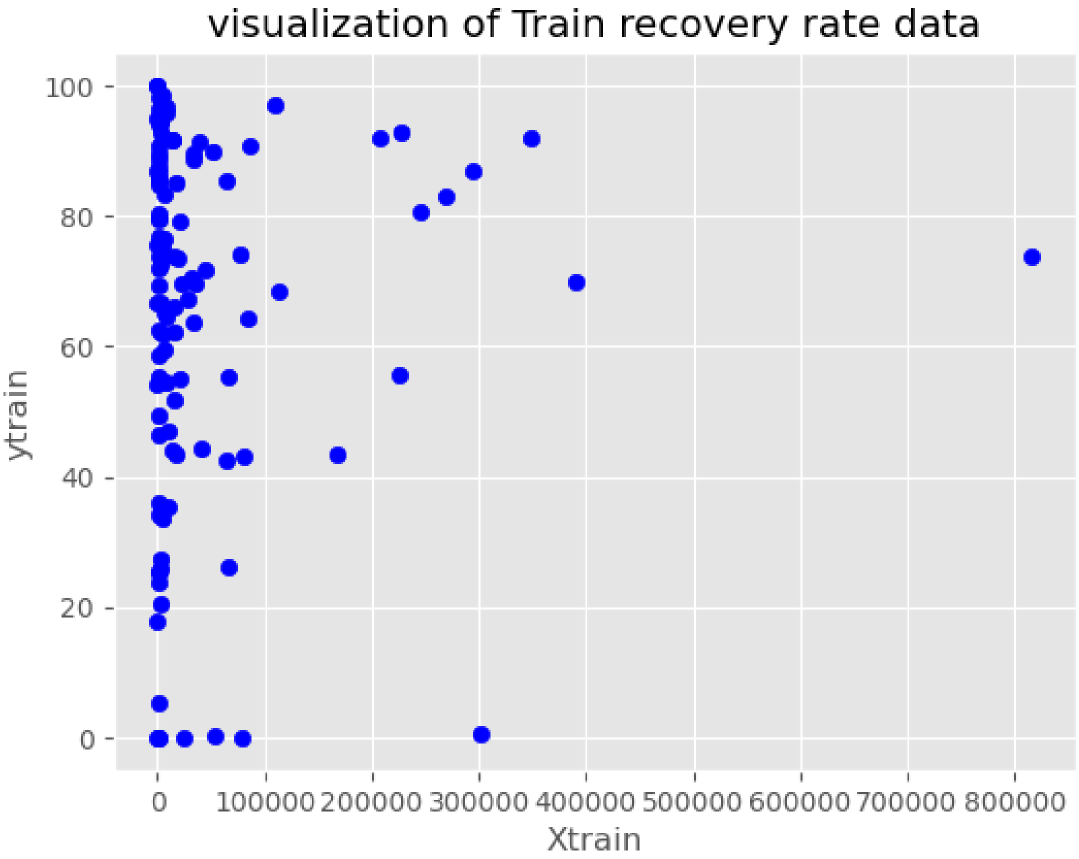
```
Text(0.5, 1.0, 'visualization of predicted recovery rate data')
```



this graph is a visualization of preducted recovery rate data with the predicted y values in the y axis and the x test values in the x axis

```
plt.scatter(X2, rec_rate_train, c='b', label='Tain data')
plt.xlabel('Xtrain')
plt.ylabel('ytrain')
plt.title('visualization of Train recovery rate data')
```

```
Text(0.5, 1.0, 'visualization of Train recovery rate data')
```

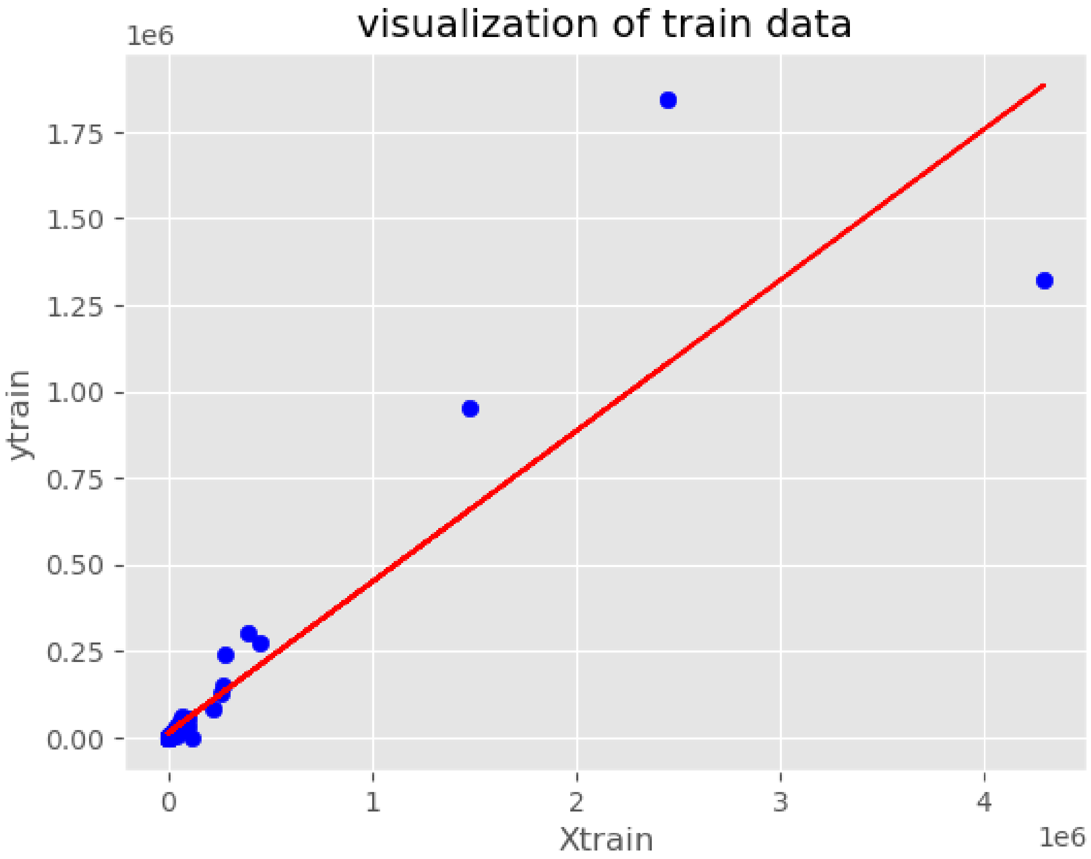


for this graph, we are visualizing a training of recovery rate data with both the y train values and the x train values

after we have looked at the few similarities between the recovery rate tested and predicted, now we will look at a plotting of the training data and the predicted data

```
plt.scatter(X1, y1, c='b', label='Train data')
plt.plot(X1, b+ slope * X1, c='red', label='train model')
plt.xlabel('Xtrain')
plt.ylabel('ytrain')
plt.title('visualization of train data')
```

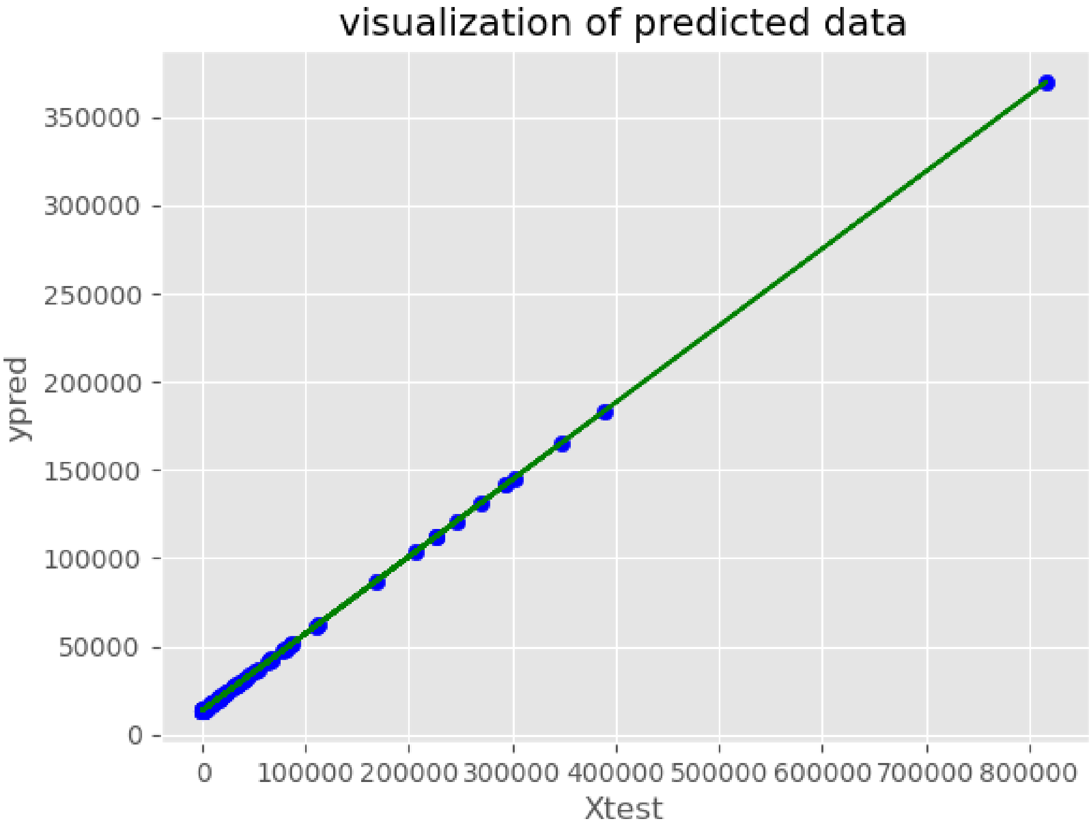
```
Text(0.5, 1.0, 'visualization of train data')
```



this graph shows another visualization of training data with plotted line using both y train and x train values

```
plt.scatter(X2, y2_model, c='b', label='predicted data')
plt.plot(X2, b+ slope * X2, c='green', label='predicted model')
plt.xlabel('Xtest')
plt.ylabel('ypred')
plt.title('visualization of predicted data')
```

```
Text(0.5, 1.0, 'visualization of predicted data')
```



this time, we are visualizing the predicted data with y predicted values and x testing values

since we didn't find a very high value of accuracy we are going to test if the linear regression is actually a multiple linear regression by having a 2D array for x taking the confirmed cases AND the active cases

```
x = data[['Confirmed','Active']].values
y = data['Recovered/']
from sklearn.model_selection import train_test_split
# split the data with 60% in each set
X1, X2, y1, y2 = train_test_split(x,y, random_state=1,train_size=0.4)
from sklearn.linear_model import LinearRegression
model = LinearRegression(fit_intercept=True)
model.fit(X1,y1)
y2_model = model.predict(X2)
from sklearn.metrics import r2_score
r2_score(y2,y2_model)
```

```
0.99660774223063
```

while testing the train sizes of the splitting, all of the cases gave us an average of 0.99 accuracy meaning that there is definitely a multiple linear regrassion between confirmed , active cases and the recovery cases

```
#define cross-validation method to use
cv = KFold(n_splits=10, random_state=1, shuffle=True)
#use k-fold CV to evaluate model
scores = cross_val_score(model, x, y,
scoring='neg_mean_absolute_error',cv=cv, n_jobs=-1)
#view mean absolute error
sqrt(mean(absolute(scores)))
```

```
50.524170511661
```

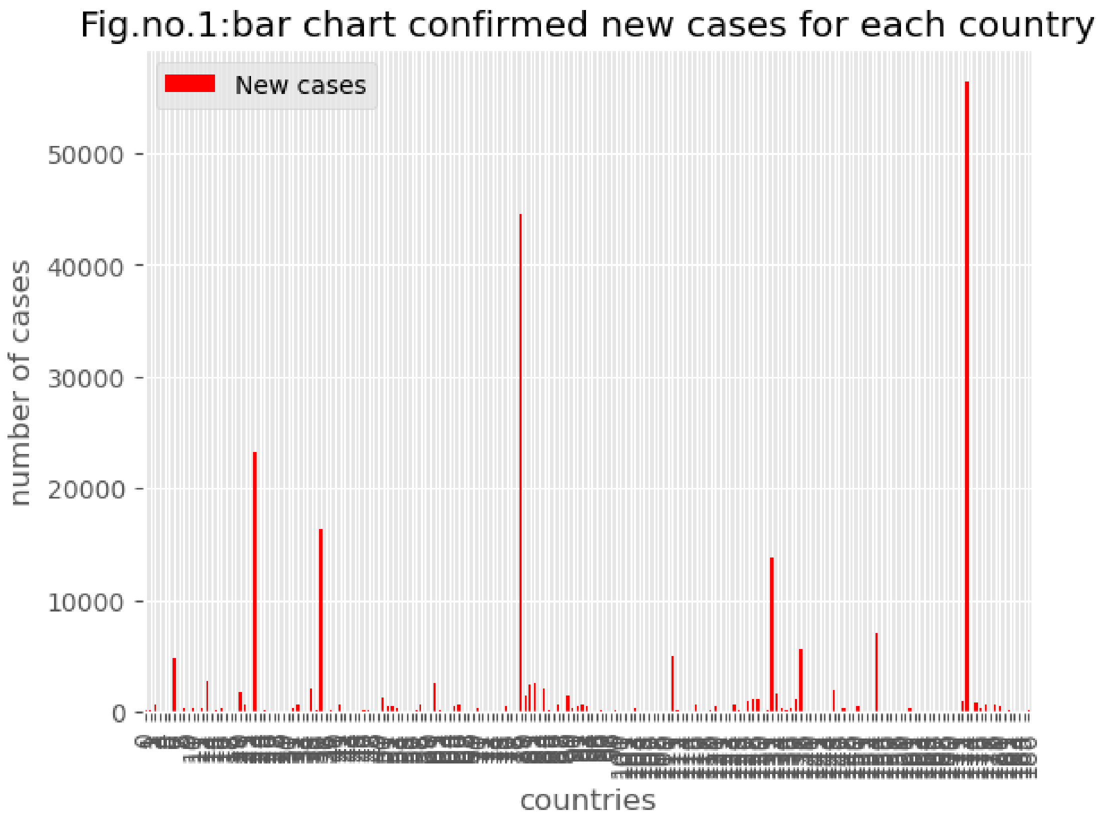
the sqrt of the MAE is lower than before nearly 3 times less errors, therefore this confirms the 0.99 accuracy we found.

Here we covered the confirmed cases and the number of new cases

```
import matplotlib.pyplot as plt
data2=pd.DataFrame(data[['Country/Region','New cases']])
plt.figure();
data2.plot.bar(title='Fig.no.1:bar chart confirmed new cases for each
country',color='red');
plt.xlabel('countries');
plt.ylabel('number of cases')
```

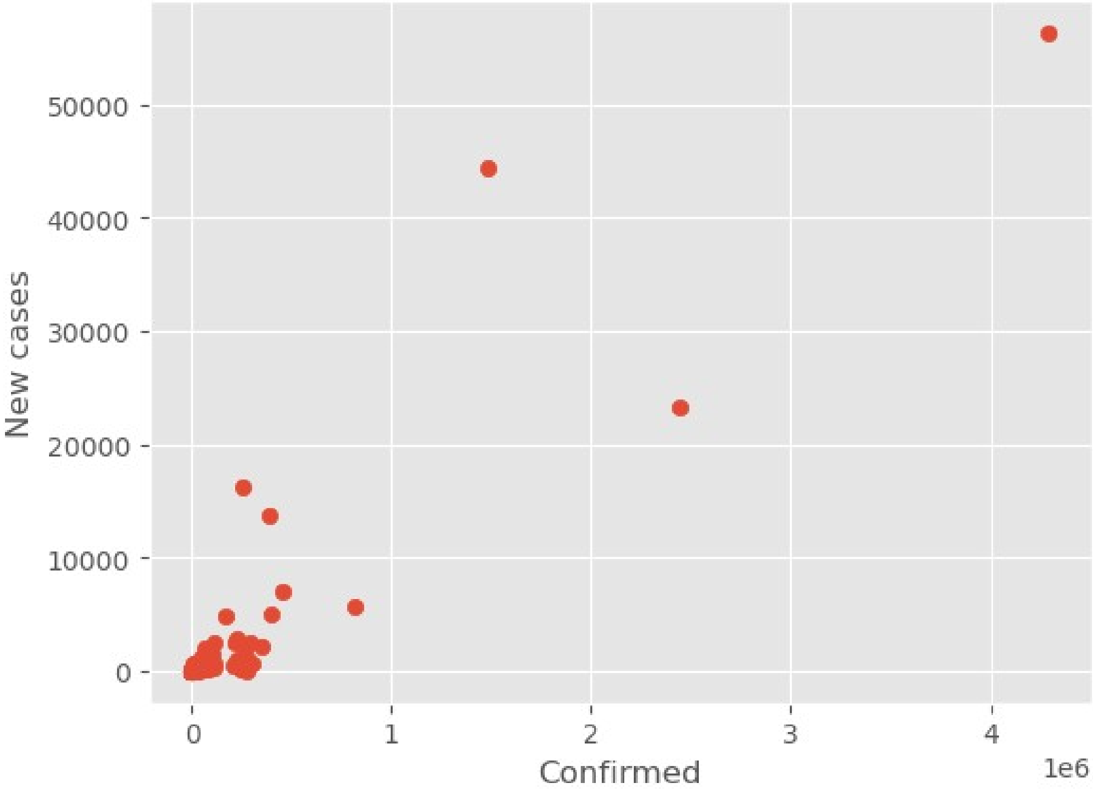
```
Text(0, 0.5, 'number of cases')
```

```
<Figure size 640x480 with 0 Axes>
```



here is a chart that shows all new cases per country

```
import matplotlib.pyplot as plt
import numpy as np
x = data['Confirmed']
y = data['New cases']
plt.xlabel('Confirmed')
plt.ylabel("New cases")
plt.scatter(x, y);
```

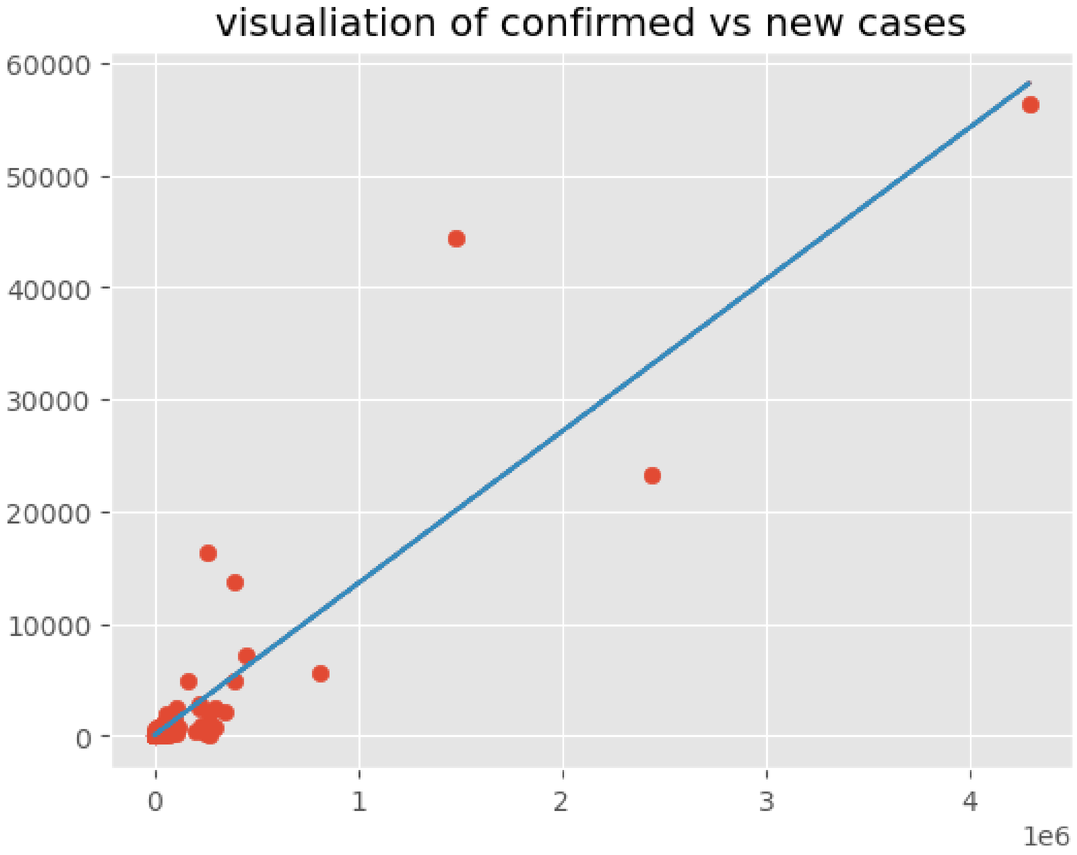


this graph represents scattered dots showing the new cases versus the confirmed cases

```
from sklearn.linear_model import LinearRegression
model = LinearRegression(fit_intercept=True)
X = x[:, np.newaxis]
model.fit(X, y)
slope=model.coef_
inter=model.intercept_
xfit = np.linspace(-1, 4290259)
Xfit = xfit[:, np.newaxis]
yfit = model.predict(Xfit)
plt.scatter(x, y)
plt.plot(Xfit,yfit);
plt.plot(x, slope*x+inter);
plt.title('visualiation of confirmed vs new cases')
```

```
<ipython-input-17-a1cd6d0e27a0>:3: FutureWarning: Support for multi-
dimensional indexing (e.g. `obj[:, None]`) is deprecated and will be
removed in a future version.  Convert to a numpy array before indexing
instead.
  X = x[:, np.newaxis]
```

```
Text(0.5, 1.0, 'visualiation of confirmed vs new cases')
```



this graph is a visualization of confirmed cases versus the new cases

```
from sklearn.model_selection import train_test_split
# split the data with 80% in each set
X1, X2, y1, y2 = train_test_split(x,y, random_state=1,train_size=0.2)
from sklearn.linear_model import LinearRegression
model = LinearRegression(fit_intercept=True)
model.fit((X1.values).reshape(-1,1),(y1.values))
slope=model.coef_
inter=model.intercept_
y2_model = model.predict((X2.values).reshape(-1,1))
from sklearn.metrics import r2_score
r2=r2_score((y2.values).reshape(-1,1),y2_model)
r2
```

```
-0.9723917937918602
```

here we did the same thing as before only this case is analyzing the confirmed cases and the new cases, the accuracy score was -97% which tells us that there is no correlation between those two, however we will still try with cross validation method.

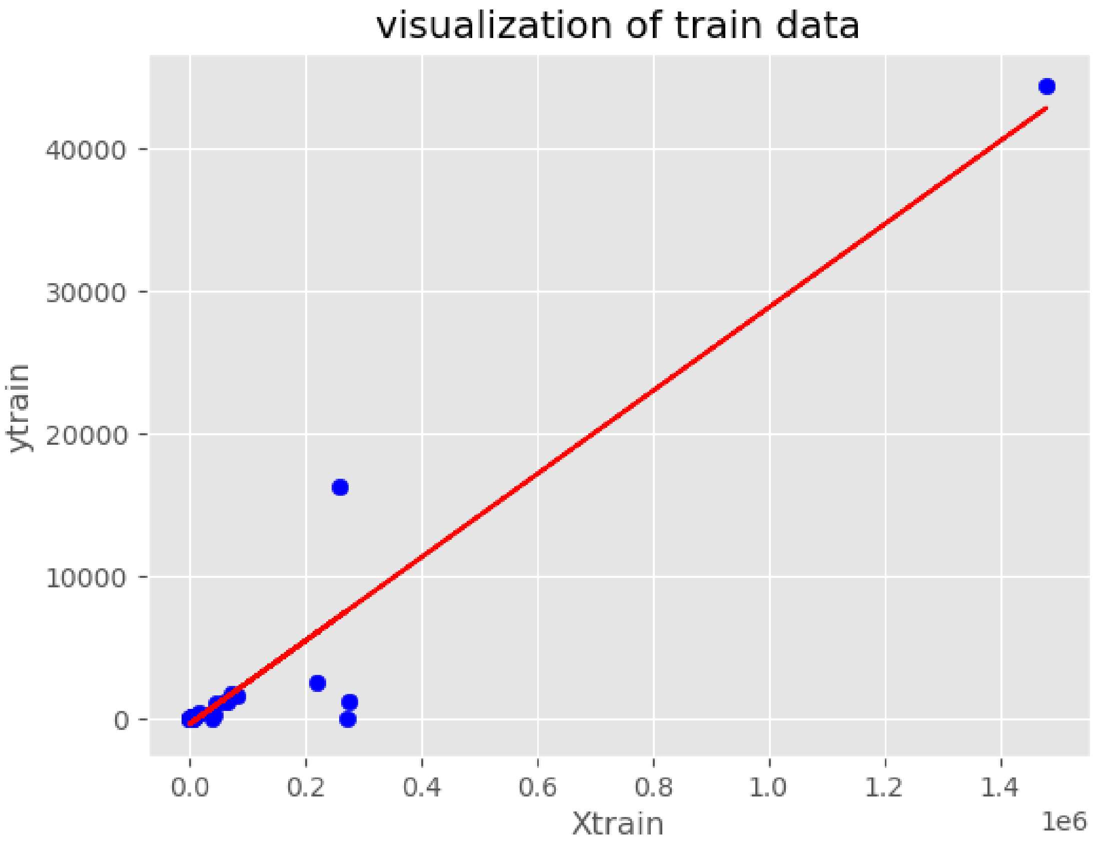
```
#define cross-validation method to use
cv = KFold(n_splits=10, random_state=1, shuffle=True)
#use k-fold CV to evaluate model
scores = cross_val_score(model, x, y,
scoring='neg_mean_absolute_error',
                         cv=cv, n_jobs=-1)

#view mean absolute error
sqrt(mean(absolute(scores)))
```

```
nan
```

the cross validation gave us a NAN value meaning that our prediction was true, there is no correlation between the two

```
plt.scatter(X1, y1, c='b', label='Train data')
plt.plot(X1, inter+ slope * X1, c='red', label='train model')
plt.xlabel('Xtrain')
plt.ylabel('ytrain')
plt.title('visualization of train data')
```

```
Text(0.5, 1.0, 'visualization of train data')
```



this graph is a visualization of training data with y train data and x train data. even the scattered dots of the train data aren't very much aligned . ssince ther is no correlation there is no need for visualizing the test data.

we wanted to try and find a multiple linear regression with confirmed, active and new cases, however the accuracy was still low about 30% which is not so high compared to earlier when we found 99%.

```
x = data[['Confirmed','Active']]
y = data['New cases']
from sklearn.model_selection import train_test_split
# split the data with 60% in each set
X1, X2, y1, y2 = train_test_split(x,y, random_state=1,train_size=0.4)
from sklearn.linear_model import LinearRegression
model = LinearRegression(fit_intercept=True)
model.fit(X1,y1)
y2_model = model.predict(X2)
from sklearn.metrics import r2_score
r2_score(y2,y2_model)
```

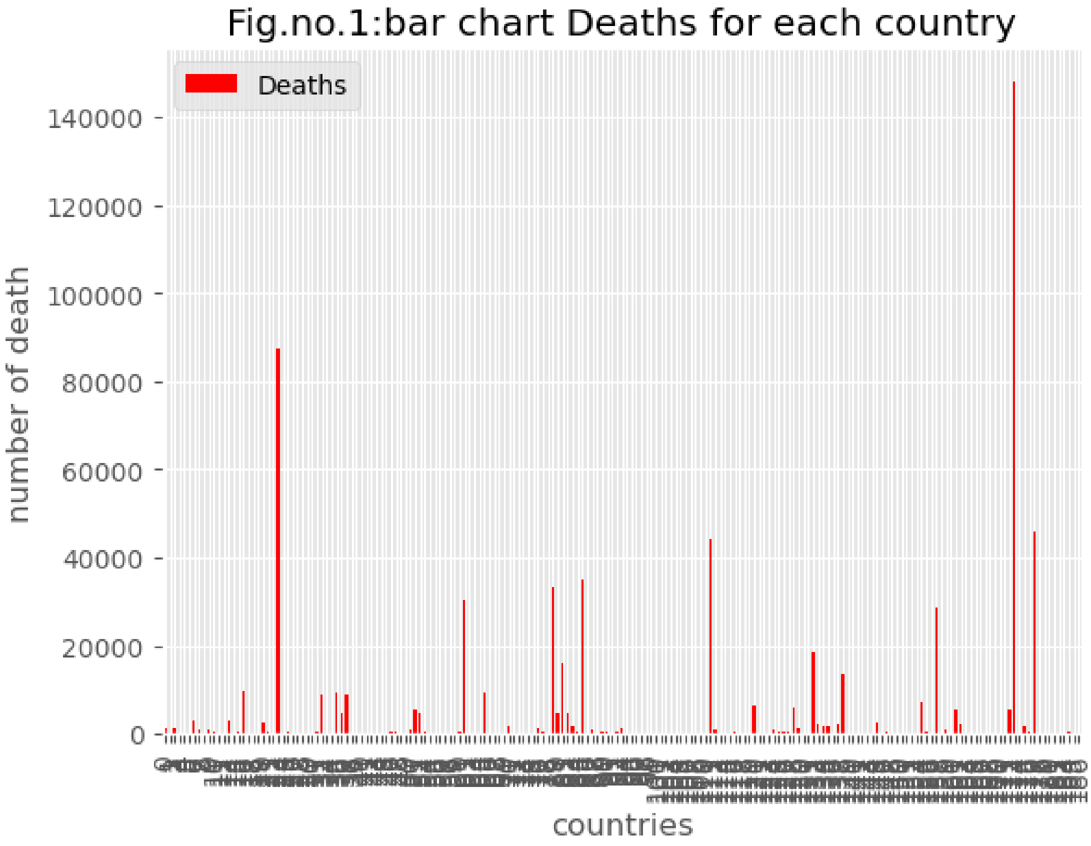
```
0.3343743100616995
```

Here we covered the confirmed and death with a comparison of the estimated death rate and the actual death rate

```
import matplotlib.pyplot as plt
data2=pd.DataFrame(data[['Country/Region','Deaths']])
plt.figure();
data2.plot.bar(title='Fig.no.1:bar chart Deaths for each
country',color='red');
plt.xlabel('countries');
plt.ylabel('number of death')
```

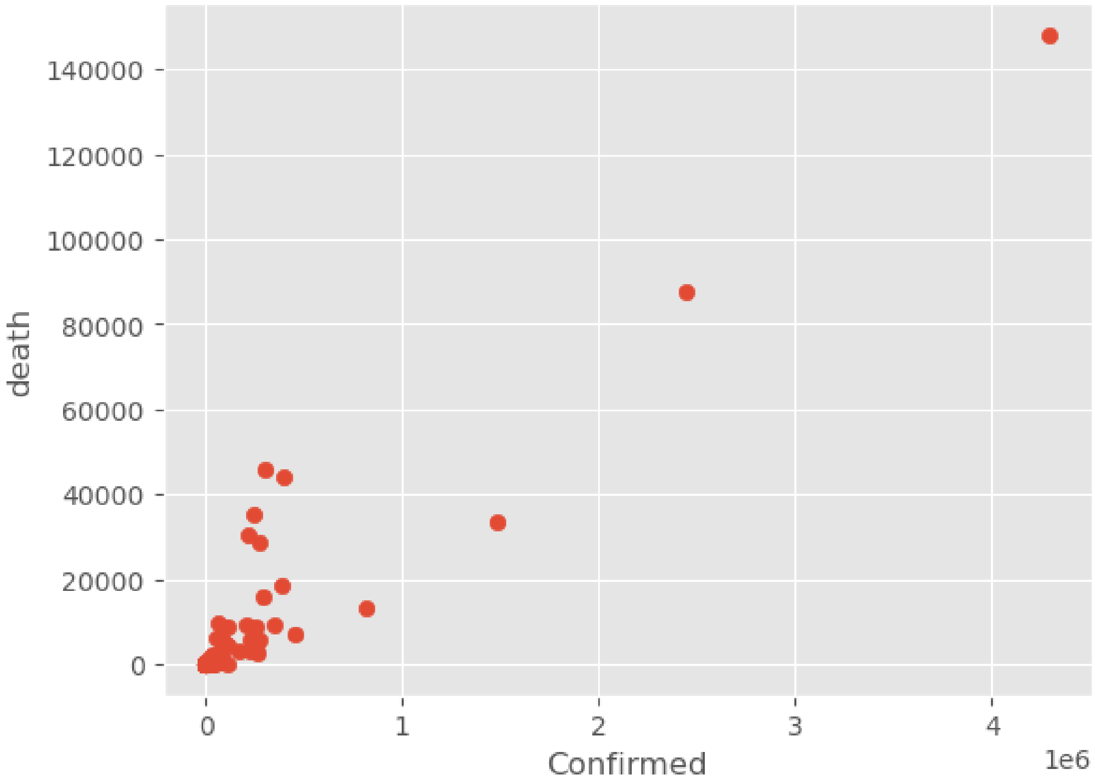
```
Text(0, 0.5, 'number of death')
```

```
<Figure size 640x480 with 0 Axes>
```



this figure shows a chart of deaths per each country

```
import matplotlib.pyplot as plt
import numpy as np
x = data['Confirmed']
y = data['Deaths']
plt.xlabel('Confirmed')
plt.ylabel("death")
plt.scatter(x, y);
```



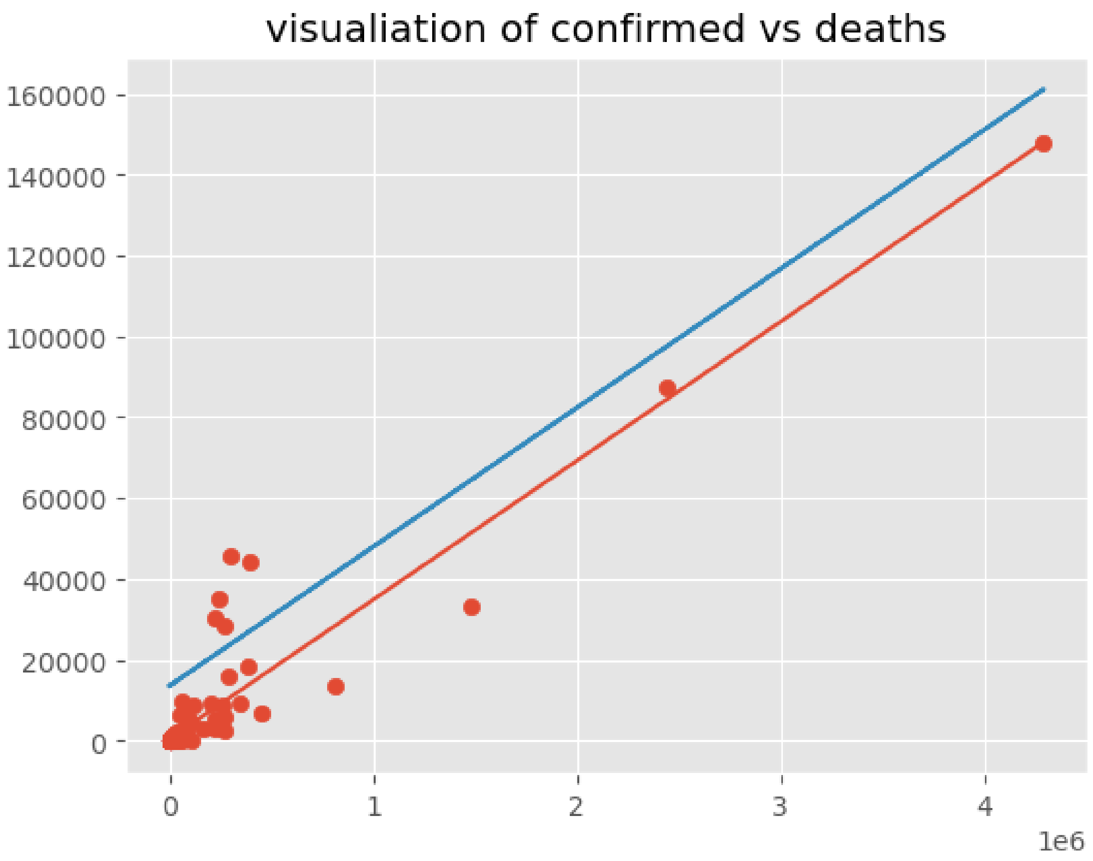
this graph represents the number of deaths versus the number of confirmed cases

```
from sklearn.linear_model import LinearRegression
model = LinearRegression(fit_intercept=True)
X = x[:, np.newaxis]
model.fit(X, y)
slope=model.coef_
inter=model.intercept_
xfit = np.linspace(-1, 4290259)
Xfit = xfit[:, np.newaxis]
yfit = model.predict(Xfit)
```

```
<ipython-input-24-be2b362c21c9>:3: FutureWarning: Support for multi-
dimensional indexing (e.g. `obj[:, None]`) is deprecated and will be
removed in a future version.  Convert to a numpy array before indexing
instead.
  X = x[:, np.newaxis]
```

```
plt.scatter(x, y)
plt.plot(Xfit,yfit);
plt.plot(x, slope*x+b);
plt.title('visualiation of confirmed vs deaths')
```

```
Text(0.5, 1.0, 'visualiation of confirmed vs deaths')
```

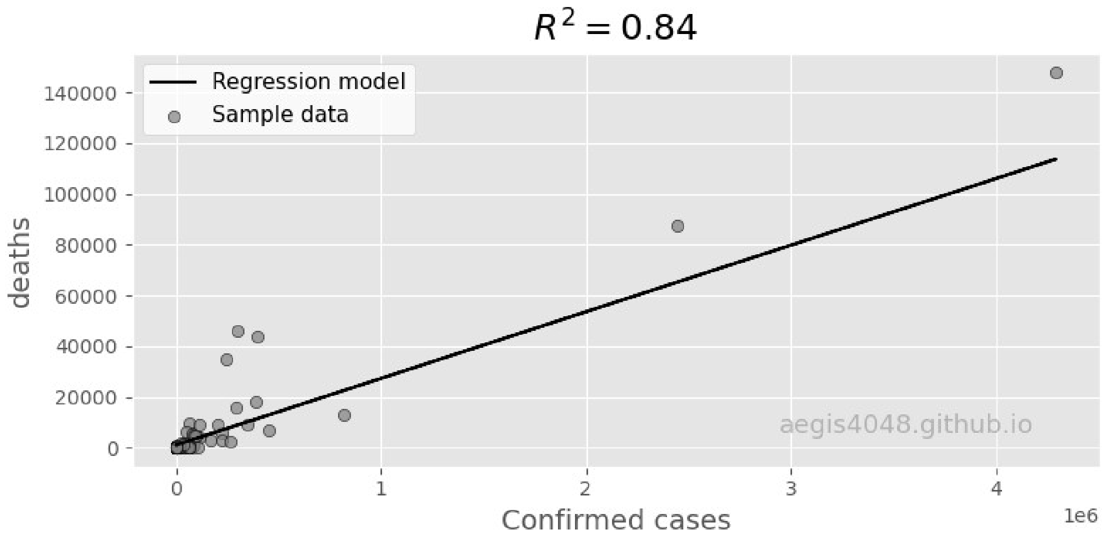


this graph shows two lines , the blue one shows the plot from x fit and y fit and the red one was plotted using the slope and the intercept from the model

```
from sklearn.model_selection import train_test_split
# split the data with 80% in each set
X1, X2, y1, y2 = train_test_split(x,y, random_state=1,train_size=0.2)
from sklearn.linear_model import LinearRegression
model = LinearRegression(fit_intercept=True)
model.fit((X1.values).reshape(-1,1),(y1.values))
slope=model.coef_
inter=model.intercept_
y2_model = model.predict((X2.values).reshape(-1,1))
from sklearn.metrics import r2_score
r2=r2_score((y2.values).reshape(-1,1),y2_model)
r2
```

```
0.8416654267896824
```

```
plt.style.use('default')
plt.style.use('ggplot')

fig, ax = plt.subplots(figsize=(8, 4))

ax.plot(X2, y2_model, color='k', label='Regression model')
ax.scatter(X2, y2, edgecolor='k', facecolor='grey', alpha=0.7,
label='Sample data')
ax.set_ylabel('deaths', fontsize=14)
ax.set_xlabel('Confirmed cases', fontsize=14)
ax.text(0.8, 0.1, 'aegis4048.github.io', fontsize=13, ha='center',
va='center',
         transform=ax.transAxes, color='grey', alpha=0.5)
ax.legend(facecolor='white', fontsize=11)
ax.set_title('$R^2= %.2f$' % r2, fontsize=18)

fig.tight_layout()
```



as we did the previous times, here we wanted to test the linear regression between confirmed cases and deaths cases, we found the accuracy score to be high enough 84% which was also the average testing between all the train sizes. we are still going to see what the cross validation will give us for the MAE.

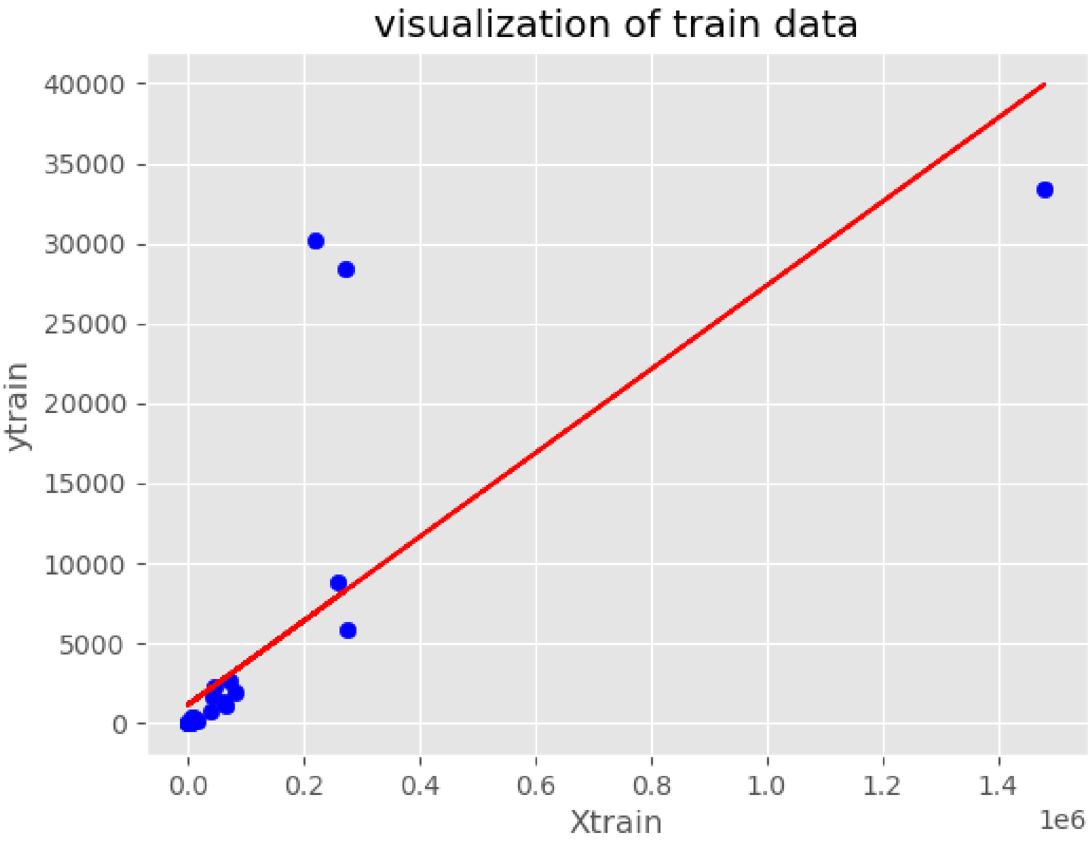
```
#define cross-validation method to use
cv = KFold(n_splits=10, random_state=1, shuffle=True)
#use k-fold CV to evaluate model
scores = cross_val_score(model, X, y,
scoring='neg_mean_absolute_error',
                         cv=cv, n_jobs=-1)

#view mean absolute error
sqrt(mean(absolute(scores)))
```

```
44.69202742906502
```

the MAE we had was about 44 which is good confirming what we said earlier.

```
plt.scatter(X1, y1, c='b', label='Train data')
plt.plot(X1,inter+ slope * X1, c='red', label='train model')
plt.xlabel('Xtrain')
plt.ylabel('ytrain')
plt.title('visualization of train data')
```

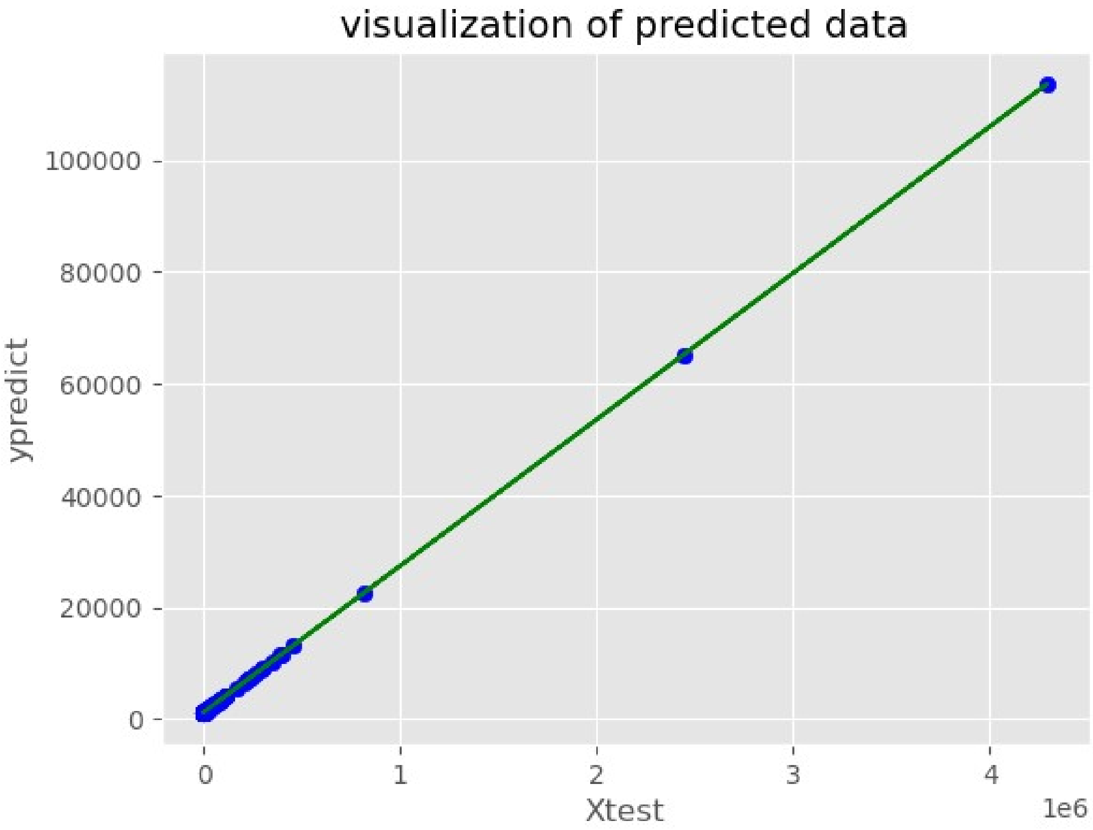
```
Text(0.5, 1.0, 'visualization of train data')
```



this graph shows a visualization of training data with the ytrain data and the x train data

```
plt.scatter(X2, y2_model, c='b', label='model data')
plt.plot(X2, inter+ slope * X2, c='green', label='train model')
plt.xlabel('Xtest')
plt.ylabel('ypredict')
plt.title('visualization of predicted data')
```

```
Text(0.5, 1.0, 'visualization of predicted data')
```



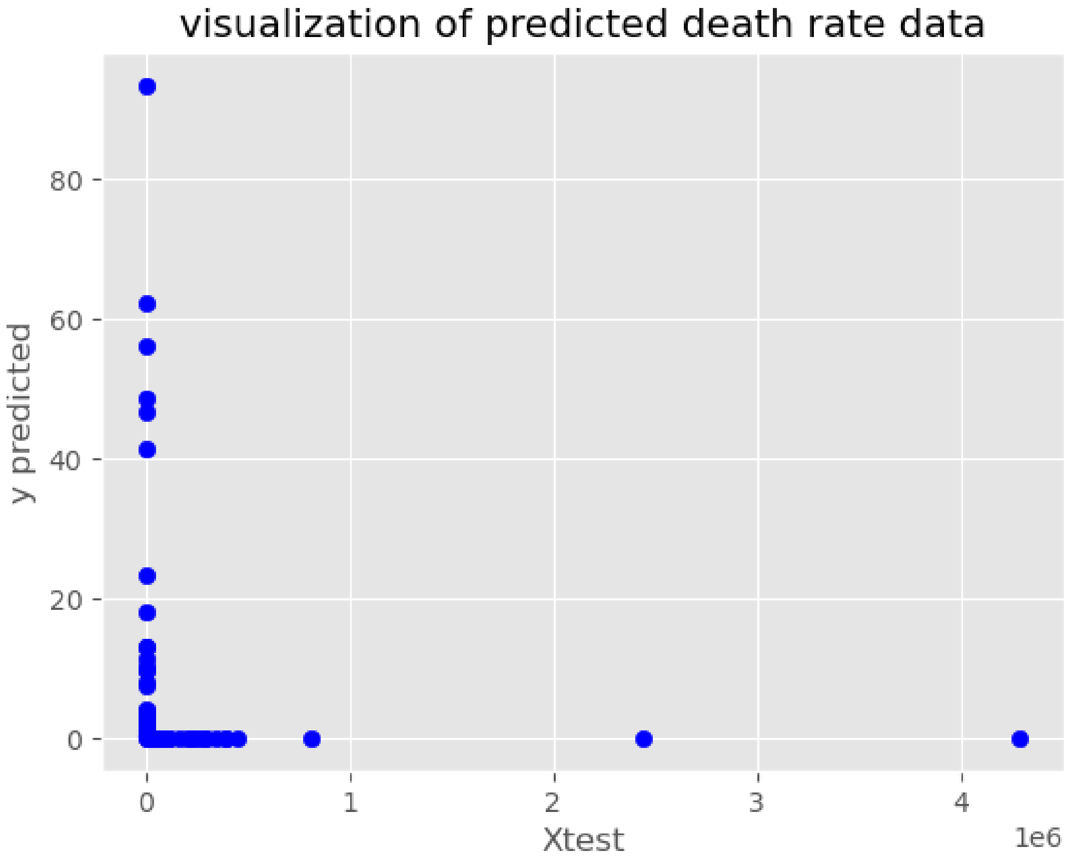
this graph is a visualization of predicted data with y predicted values and x tested values.

```
death_rate_model=y2_model/X2
death_rate_train=y2*100/X2
```

as we did for the recovery rate we will do the same for the death rate which will also show similarities in the plotting.

```
plt.scatter(X2, death_rate_model, c='b', label='Predicted data')
plt.xlabel('Xtest')
plt.ylabel('y predicted')
plt.title('visualization of predicted death rate data')
```

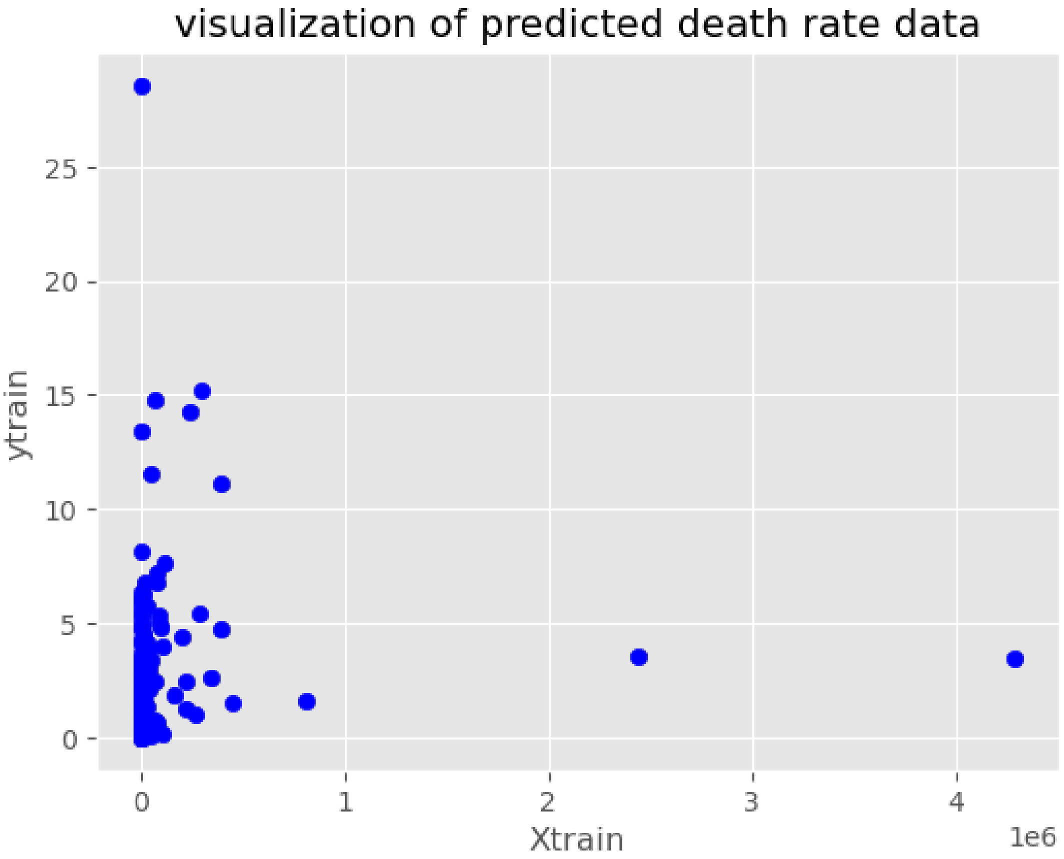
```
Text(0.5, 1.0, 'visualization of predicted death rate data')
```



this graph shows a visualization of predicted death rate data using y predicted values and x testing values

```
plt.scatter(X2, death_rate_train, c='b', label='Train data')
plt.xlabel('Xtrain')
plt.ylabel('ytrain')
plt.title('visualization of predicted death rate data')
```

```
Text(0.5, 1.0, 'visualization of predicted death rate data')
```



this graph shows a visualization of predicted death rate data using y training values and x training values

```
x = data[['Confirmed','Active']]
y = data['Deaths']
from sklearn.model_selection import train_test_split
# split the data with 60% in each set
X1, X2, y1, y2 = train_test_split(x,y, random_state=1,train_size=0.4)
from sklearn.linear_model import LinearRegression
model = LinearRegression(fit_intercept=True)
model.fit(X1,y1)
y2_model = model.predict(X2)
from sklearn.metrics import r2_score
r2_score(y2,y2_model)
```

```
0.40447509128633086
```

```
#define cross-validation method to use
cv = KFold(n_splits=10, random_state=1, shuffle=True)
#use k-fold CV to evaluate model
scores = cross_val_score(model, X, y,
scoring='neg_mean_absolute_error',
                         cv=cv, n_jobs=-1)
```

```
#view mean absolute error
mean(absolute(scores))
```

```
1997.3773157203002
```