2306AML117 MAHESH BABU M assignment 8

In [221…
```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
```

In [ ]:

In [222…
```python
olympics=pd.read_csv('teams.csv')
```

In [223…
```python
olympics
```

Out[223]:

	team	year	athletes	events	age	height	weight	prev_medals	medals
0	AFG	1964	8	8	22.0	161.0	64.2	0	0
1	AFG	1968	5	5	23.2	170.2	70.0	0	0
2	AFG	1972	8	8	29.0	168.3	63.8	0	0
3	AFG	1980	11	11	23.6	168.4	63.2	0	0
4	AFG	2004	5	5	18.6	170.8	64.8	0	0
...	...	...	...	...	...	...	...	...	...
2009	ZIM	2000	26	19	25.0	179.0	71.1	0	0
2010	ZIM	2004	14	11	25.1	177.8	70.5	0	3
2011	ZIM	2008	16	15	26.1	171.9	63.7	3	4
2012	ZIM	2012	9	8	27.3	174.4	65.2	4	0
2013	ZIM	2016	31	13	27.5	167.8	62.2	0	0

2014 rows × 9 columns

In [224…
```python
olympics.isnull().sum()
```

Out[224]:
```
team           0
year           0
athletes       0
events         0
age            0
height         0
weight         0
prev_medals    0
medals         0
dtype: int64
```

In [225…
```python
print(olympics.dtypes)
```

```
team            object
year             int64
athletes         int64
events           int64
age            float64
height         float64
weight         float64
prev_medals      int64
medals           int64
dtype: object
```

In [226…
```python
print(olympics.describe())
```

```
              year     athletes       events          age       height  \
count  2014.000000  2014.000000  2014.000000  2014.000000  2014.000000   
mean   1995.227408    76.329692    36.877855    24.812612   173.955164   
std      15.227727   129.799427    50.130877     2.758258     5.262469   
min    1964.000000     1.000000     1.000000    17.000000   151.000000   
25%    1984.000000     7.000000     6.000000    23.300000   170.600000   
50%    1996.000000    21.000000    14.000000    24.700000   174.400000   
75%    2008.000000    74.750000    47.000000    26.100000   177.300000   
max    2016.000000   839.000000   270.000000    66.000000   193.000000   

            weight  prev_medals       medals  
count  2014.000000  2014.000000  2014.000000  
mean     69.328997    10.248759    10.990070  
std       7.494740    31.951920    33.627528  
min      43.300000     0.000000     0.000000  
25%      64.700000     0.000000     0.000000  
50%      69.500000     0.000000     0.000000  
75%      73.400000     4.000000     5.000000  
max     148.000000   442.000000   442.000000  
```

In [227…
```python
olympics_1 = olympics.sort_values(['year','athletes'],ascending = [True,False])\
                        .reset_index(drop = True)
print(olympics_1.shape)
olympics_1.head(10)
```

```
(2014, 9)
```

Out[227]:

	team	year	athletes	events	age	height	weight	prev_medals	medals
0	USA	1964	492	159	23.5	177.2	71.8	125	169
1	GER	1964	478	159	24.9	174.8	71.2	89	116
2	URS	1964	463	154	26.0	174.7	72.8	169	174
3	JPN	1964	456	155	24.3	167.7	63.6	31	62
4	AUS	1964	392	133	24.4	174.5	70.2	46	44
5	HUN	1964	319	111	24.5	173.8	70.4	66	56
6	GBR	1964	301	124	25.0	175.0	69.9	28	28
7	ITA	1964	256	91	25.0	175.0	72.6	88	51
8	KOR	1964	240	93	23.4	168.2	62.9	0	3
9	POL	1964	240	87	25.7	172.6	69.2	30	46

In [ ]:

In [ ]:

In [ ]:

In [228…
```python
top_25 = olympics_1.drop(['year'],axis=1).sort_values(['medals'])
top_25.head()
```

Out[228]:

	team	athletes	events	age	height	weight	prev_medals	medals
1006	SUD	4	4	25.0	172.0	63.3	0	0
1177	SKN	4	4	25.0	174.0	71.0	0	0
1175	MON	4	4	25.8	177.5	71.2	0	0
1174	MDV	4	4	18.2	160.5	47.5	0	0
1173	IRQ	4	4	24.5	164.8	57.8	0	0

In [229…
```python
olympics1 = olympics_1.groupby(['team']).sum().sort_values(['medals'], ascending = False).reset_index()
olympics1
```

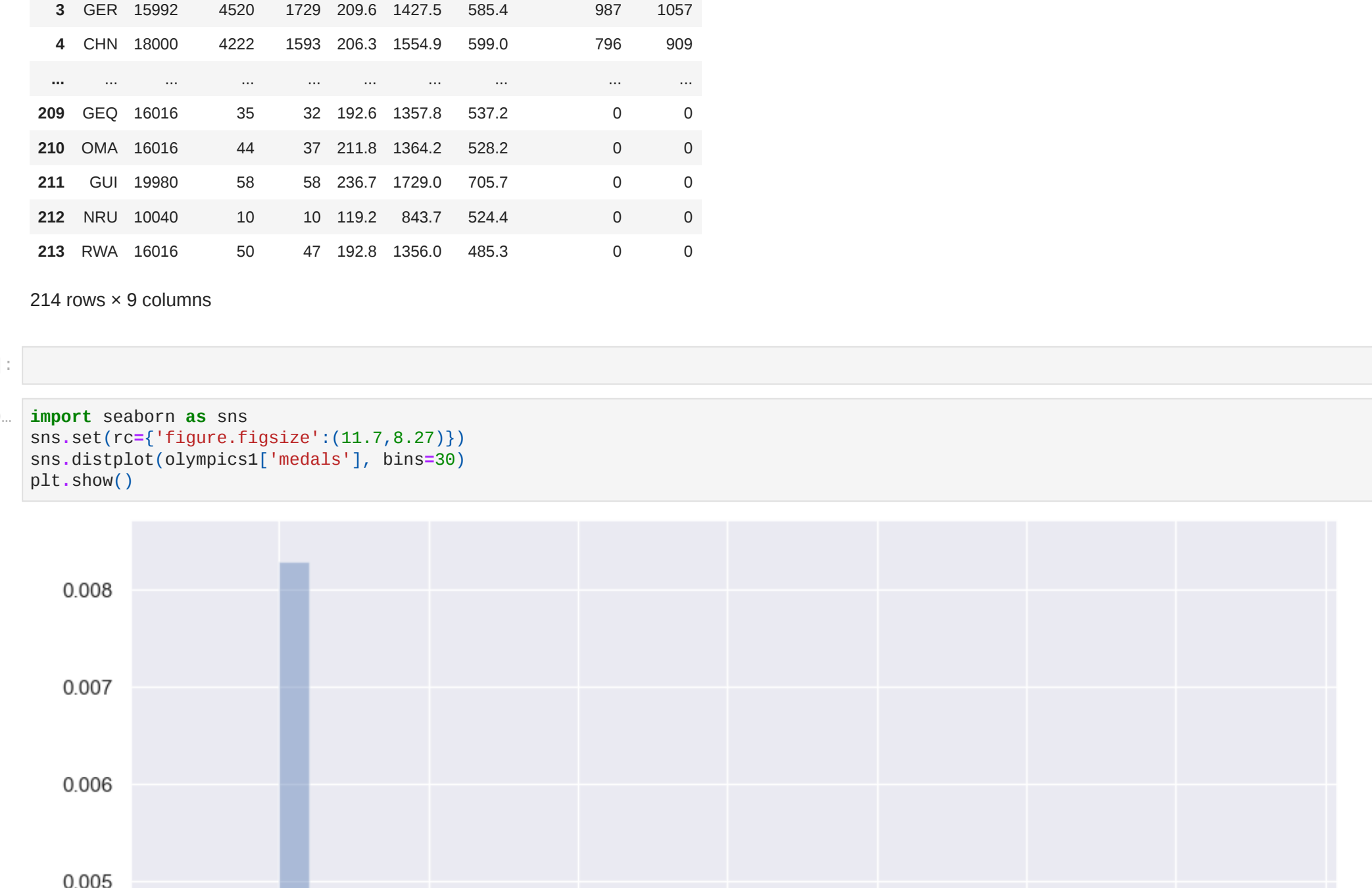
Out[229]:

	team	year	athletes	events	age	height	weight	prev_medals	medals
0	USA	25880	8795	2925	328.1	2306.4	944.8	2907	3046
1	URS	11848	3346	1110	147.1	1054.9	436.2	1477	1608
2	AUS	27860	5953	2311	345.3	2478.6	1011.8	1076	1112
3	GER	15992	4520	1729	209.6	1427.5	585.4	987	1057
4	CHN	18000	4222	1593	206.3	1554.9	599.0	796	909
...	...	...	...	...	...	...	...	...	...
209	GEQ	16016	35	32	192.6	1357.8	537.2	0	0
210	OMA	16016	44	37	211.8	1364.2	528.2	0	0
211	GUI	19980	58	58	236.7	1729.0	705.7	0	0
212	NRU	10040	10	10	119.2	843.7	524.4	0	0
213	RWA	16016	50	47	192.8	1356.0	485.3	0	0

214 rows × 9 columns

In [ ]:

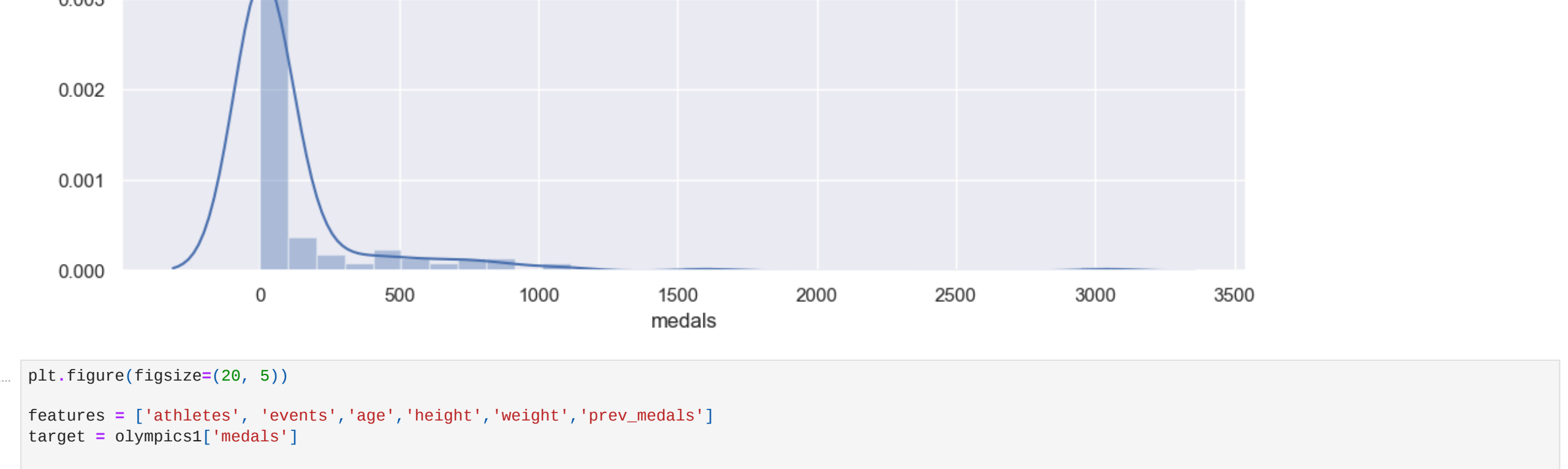
In [230…
```python
import seaborn as sns
sns.set(rc={'figure.figsize':(11.7,8.27)})
sns.distplot(olympics1['medals'], bins=30)
plt.show()
```



In [231…
```python
plt.figure(figsize=(20, 5))

features = ['athletes', 'events','age','height','weight','prev_medals']
target = olympics1['medals']

for i, col in enumerate(features):
    plt.subplot(1, len(features) , i+1)
    x = olympics[col]
    y =olympics['medals']
    plt.scatter(x, y, marker='o')
    plt.title(col)
    plt.xlabel(col)
    plt.ylabel('medals')
```



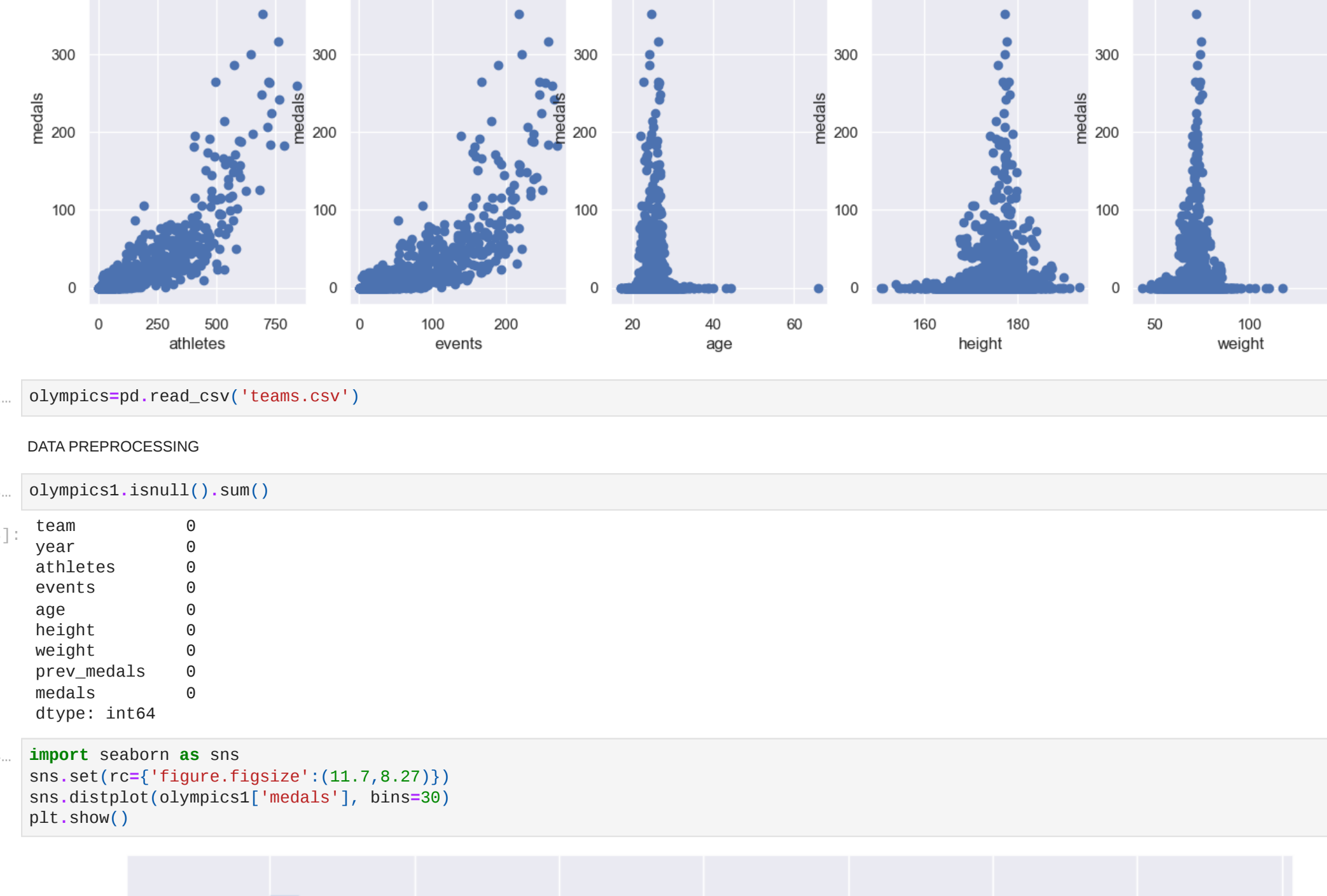
In [232…
```python
olympics=pd.read_csv('teams.csv')
```

DATA PREPROCESSING

In [233…
```python
olympics1.isnull().sum()
```

Out[233]:
```
team           0
year           0
athletes       0
events         0
age            0
height         0
weight         0
prev_medals    0
medals         0
dtype: int64
```

In [234…
```python
import seaborn as sns
sns.set(rc={'figure.figsize':(11.7,8.27)})
sns.distplot(olympics1['medals'], bins=30)
plt.show()
```



In [ ]:

TRAINING THE MODEL and data split

In [235…
```python
X = pd.DataFrame(olympics1)
Y = olympics1['medals']
```

In [236…
```python
X
```

Out[236]:

	team	year	athletes	events	age	height	weight	prev_medals	medals
0	USA	25880	8795	2925	328.1	2306.4	944.8	2907	3046
1	URS	11848	3346	1110	147.1	1054.9	436.2	1477	1608
2	AUS	27860	5953	2311	345.3	2478.6	1011.8	1076	1112
3	GER	15992	4520	1729	209.6	1427.5	585.4	987	1057
4	CHN	18000	4222	1593	206.3	1554.9	599.0	796	909
...	...	...	...	...	...	...	...	...	...
209	GEQ	16016	35	32	192.6	1357.8	537.2	0	0
210	OMA	16016	44	37	211.8	1364.2	528.2	0	0
211	GUI	19980	58	58	236.7	1729.0	705.7	0	0
212	NRU	10040	10	10	119.2	843.7	524.4	0	0
213	RWA	16016	50	47	192.8	1356.0	485.3	0	0

214 rows × 9 columns

In [237…
```python
X1=olympics1.drop('team',axis=1)
X1
```

Out[237]:

	year	athletes	events	age	height	weight	prev_medals	medals
0	25880	8795	2925	328.1	2306.4	944.8	2907	3046
1	11848	3346	1110	147.1	1054.9	436.2	1477	1608
2	27860	5953	2311	345.3	2478.6	1011.8	1076	1112
3	15992	4520	1729	209.6	1427.5	585.4	987	1057
4	18000	4222	1593	206.3	1554.9	599.0	796	909
...	...	...	...	...	...	...	...	...
209	16016	35	32	192.6	1357.8	537.2	0	0
210	16016	44	37	211.8	1364.2	528.2	0	0
211	19980	58	58	236.7	1729.0	705.7	0	0
212	10040	10	10	119.2	843.7	524.4	0	0
213	16016	50	47	192.8	1356.0	485.3	0	0

214 rows × 8 columns

In [238…
```python
Y
```

Out[238]:
```
0      3046
1      1608
2      1112
3      1057
4       909
       ... 
209       0
210       0
211       0
212       0
213       0
Name: medals, Length: 214, dtype: int64
```

In [274…
```python
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test =  train_test_split(X1, Y, test_size = 0.3)
print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)
```

```
(149, 8)
(65, 8)
(149,)
(65,)
```

MODEL SELECTION

In [275…
```python
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

lin_model = LinearRegression()
lin_model.fit(X_train, Y_train)
```

Out[275]:

▾ LinearRegression

LinearRegression()

MODEL EVALUATION

In [276…
```python
from sklearn.metrics import r2_score
```

In [277…
```python
y_test_predict = lin_model.predict(X_test)

rmse = (np.sqrt(mean_squared_error(Y_test, y_test_predict)))
r2 = r2_score(Y_test, y_test_predict)

print('RMSE is {}'.format(rmse))
print('R2 score is {}'.format(r2))
```

```
RMSE is 8.726399543121753e-12
R2 score is 1.0
```

model2

In [278…
```python
X2 = pd.DataFrame(olympics['height'],olympics['prev_medals'], columns = ['height'])
Y2 = olympics['medals']
```

In [279…
```python
X2
```

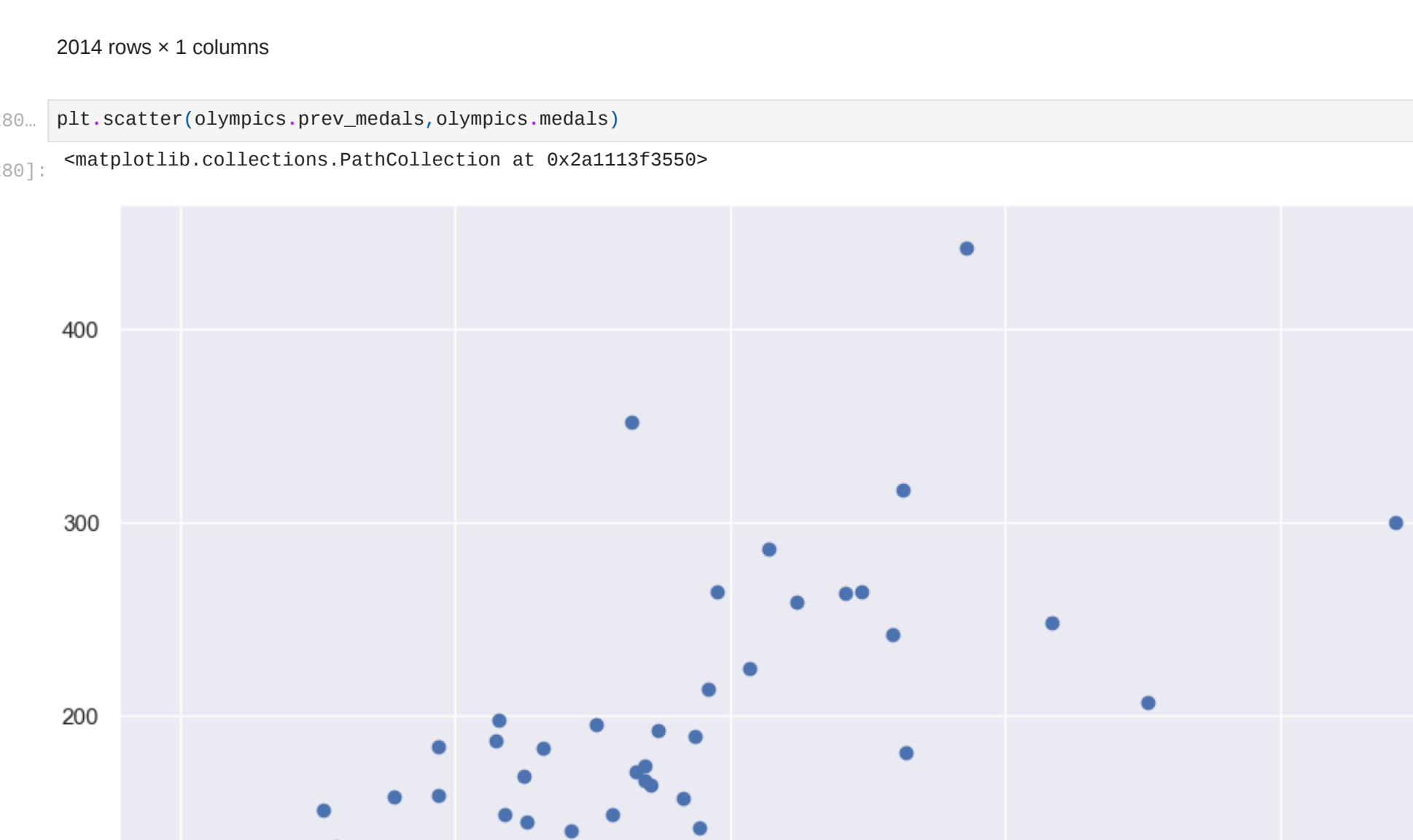
Out[279]:

	height
prev_medals	
0	161.0
0	161.0
0	161.0
0	161.0
0	161.0
...	...
0	161.0
0	161.0
3	168.4
4	170.8
0	161.0

2014 rows × 1 columns

In [280…
```python
plt.scatter(olympics.prev_medals,olympics.medals)
```

Out[280]:
```
<matplotlib.collections.PathCollection at 0x2a1113f3550>
```



In [281…
```python
Y2
```

Out[281]:
```
0       0
1       0
2       0
3       0
4       0
       ..
2009    0
2010    3
2011    4
2012    0
2013    0
Name: medals, Length: 2014, dtype: int64
```

In [282…
```python
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test =  train_test_split(X2, Y2, test_size = 0.4)
print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)
```

```
(1208, 1)
(806, 1)
(1208,)
(806,)
```

In [248…
```python
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

lin_model = LinearRegression()
lin_model.fit(X_train, Y_train)
```

Out[248]:

▾ LinearRegression

LinearRegression()

In [249…
```python
from sklearn.metrics import r2_score
```

In [250…
```python
y_test_predict = lin_model.predict(X_test)

rmse = (np.sqrt(mean_squared_error(Y_test, y_test_predict)))
r2 = r2_score(Y_test, y_test_predict)

print('RMSE is {}'.format(rmse))
print('R2 score is {}'.format(r2))
```

```
RMSE is 31.949238300307293
R2 score is 0.1759005375080599
```

In [251…
```python
from sklearn.neighbors import KNeighborsClassifier

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(X1,Y,test_size=0.3)
```

In [252…
```python
x_train
y_train
x_test
y_test
```

Out[252]:
```
57      38
166      0
194      0
128      1
80      14
      ... 
26     200
163      0
73      17
125      1
147      0
Name: medals, Length: 65, dtype: int64
```

In [253…
```python
#Fitting Classifier to the Training set
from sklearn.neighbors import KNeighborsClassifier
clf = KNeighborsClassifier()
clf.fit(x_train,y_train)
```

Out[253]:

▾ KNeighborsClassifier

KNeighborsClassifier()

In [254…
```python
#Predict on test data
y_pred=clf.predict(x_test)
```

In [255…
```python
print(y_test.values)
```

```
[  38    0    0    1   14    3  791    0  596    0  634   14    9   17
    0    2  101    0    0  546    4   23    2   93    5    1    2  196
   44   75    0    0   58   30    0   13    0    1    1 1608   19    0
    0    6    2    0    0  453   16   22   26    1   14    1  707    0
    0    0    0   17  200    0   17    1    0]
```

In [256…
```python
print(y_pred)
```

```
[  1   0   0   0   0   0   0   0  76   0 310   0   1   8   0   1  31   0
   0 200   0   1   0   7   0   0   0   0  23   8   0   0  30   4   0   0
   0   0   0  48   8   0   0   0   0   0   0   0   8   2   2   0   8   0
 440   0   0   8   0   0   4   0   0   0   0]
```

In [257…
```python
from sklearn.metrics import accuracy_score, recall_score, roc_auc_score, confusion_matrix

print("\nAccuracy score: %f" %(accuracy_score(y_test,y_pred) * 100))
```

```
Accuracy score: 30.769231
```

In [258…
```python
print(confusion_matrix(y_test, y_pred))
```

```
[[20  0  0 ...  0  0  0]
 [ 7  0  0 ...  0  0  0]
 [ 3  1  0 ...  0  0  0]
 ...
 [ 0  0  0 ...  0  0  0]
 [ 1  0  0 ...  0  0  0]
 [ 0  0  0 ...  0  0  0]]
```

In [259…
```python
#sklearn provides a tool MinMaxScaler that will scale down all the features between 0 and 1. Mathematical formula for MinMaxScaler is.

from sklearn.preprocessing import MinMaxScaler

min_max=MinMaxScaler()

X_train_minmax=min_max.fit_transform(X_train)
print(X_train_minmax)

X_test_minmax=min_max.fit_transform(X_test)

knn=KNeighborsClassifier(n_neighbors=4)

knn.fit(X_train_minmax,Y_train)

print(accuracy_score(Y_test,knn.predict(X_test_minmax)))

print('standardized')
```

```
[[0.      ]
 [0.334375]
 [0.      ]
 ...
 [0.      ]
 [0.      ]
 [0.23125 ]]
0.6004962779156328
standardized
```

In [260…
```python
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedKFold
from sklearn.linear_model import Ridge
import pandas as pd
import numpy as np

ridge_model = Ridge(alpha=1.0)
cv = RepeatedKFold(n_splits=10, n_repeats=3, random_state=1)

scores = cross_val_score(ridge_model, X1, Y, scoring='neg_mean_absolute_error', cv=cv, n_jobs=-1)

scores = np.absolute(scores)
print('Mean MAE: %.3f (%.3f)' % (np.mean(scores), np.std(scores)))
```

```
Mean MAE: 0.000 (0.000)
```

In [261…
```python
ridge_model = Ridge(alpha=0.5)

cv = RepeatedKFold(n_splits=10, n_repeats=3, random_state=1)

scores = cross_val_score(ridge_model, X1, Y, scoring='neg_mean_absolute_error', cv=cv, n_jobs=-1)

scores = np.absolute(scores)
print('Mean MAE: %.3f (%.3f)' % (np.mean(scores), np.std(scores)))
```

```
Mean MAE: 0.000 (0.000)
```

In [262…
```python
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
```

In [263…
```python
X_train, X_test, Y_train, Y_test =  train_test_split(X1, Y, test_size = 0.3)

folds=10
metric  = "neg_mean_squared_error"

models = {}
models["Linear"]        = LinearRegression()
models["Lasso"]         = Lasso()
models['Ridge']         = Ridge()

model_results = []
model_names   = []
for model_name in models:
        model   = models[model_name]
        k_fold  = KFold(n_splits=folds)
        results = cross_val_score(model, X_train, Y_train, cv=k_fold, scoring=metric)

        model_results.append(results)
        model_names.append(model_name)
        print("{}: {}, {}".format(model_name, round(results.mean(), 2), round(results.std(), 2)))
```

```
Linear: -0.0, 0.0
Lasso: -9.78, 13.63
Ridge: -0.0, 0.0
```

In [265…
```python
from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV

ridge=Ridge()
parameters={'alpha':[1e-15,1e-10,1e-8,1e-3,1e-2,1,5,10,20,30,35,40,45,50,55,100]}
ridge_regressor=GridSearchCV(ridge,parameters,scoring='neg_mean_squared_error',cv=5)
ridge_regressor.fit(X1,Y)
```

Out[265]:

▸ GridSearchCV

▸ estimator: Ridge

▸ Ridge

In [268…
```python
from sklearn.linear_model import Lasso
from sklearn.model_selection import GridSearchCV
lasso=Lasso()
parameters={'alpha':[1e-15,1e-10,1e-8,1e-3,1e-2,1,5,10,20,30,35,40,45,50,55,100]}
lasso_regressor=GridSearchCV(lasso,parameters,scoring='neg_mean_squared_error',cv=5)

lasso_regressor.fit(X1,Y)
print(lasso_regressor.best_params_)
print(lasso_regressor.best_score_)
```

```
{'alpha': 1e-15}
-0.3722719782655418
```

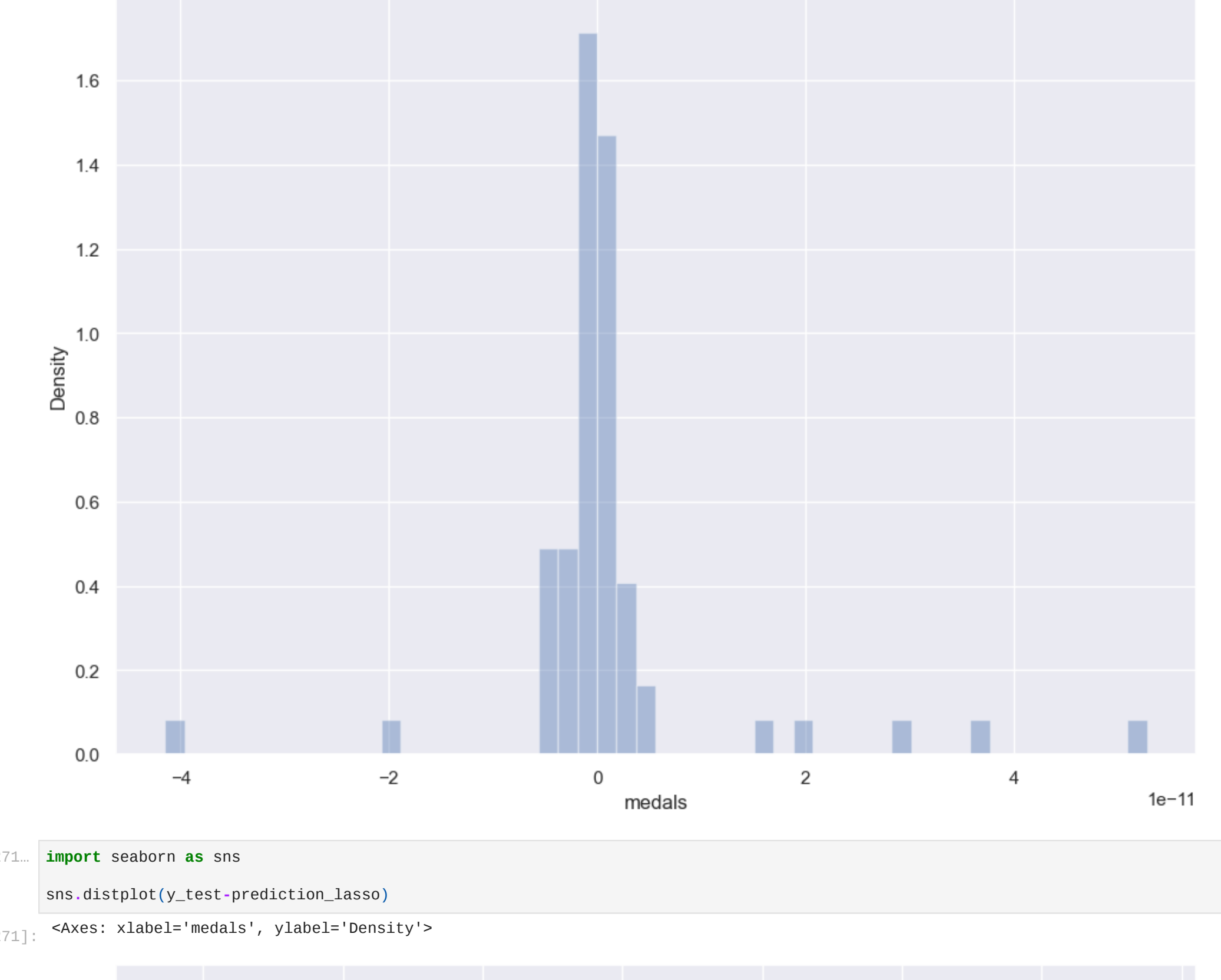
In [269…
```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X1, Y, test_size=0.3, random_state=0)

prediction_lasso=lasso_regressor.predict(X_test)
prediction_ridge=ridge_regressor.predict(X_test)
```

In [270…
```python
import seaborn as sns

sns.distplot(y_test-prediction_ridge)
```

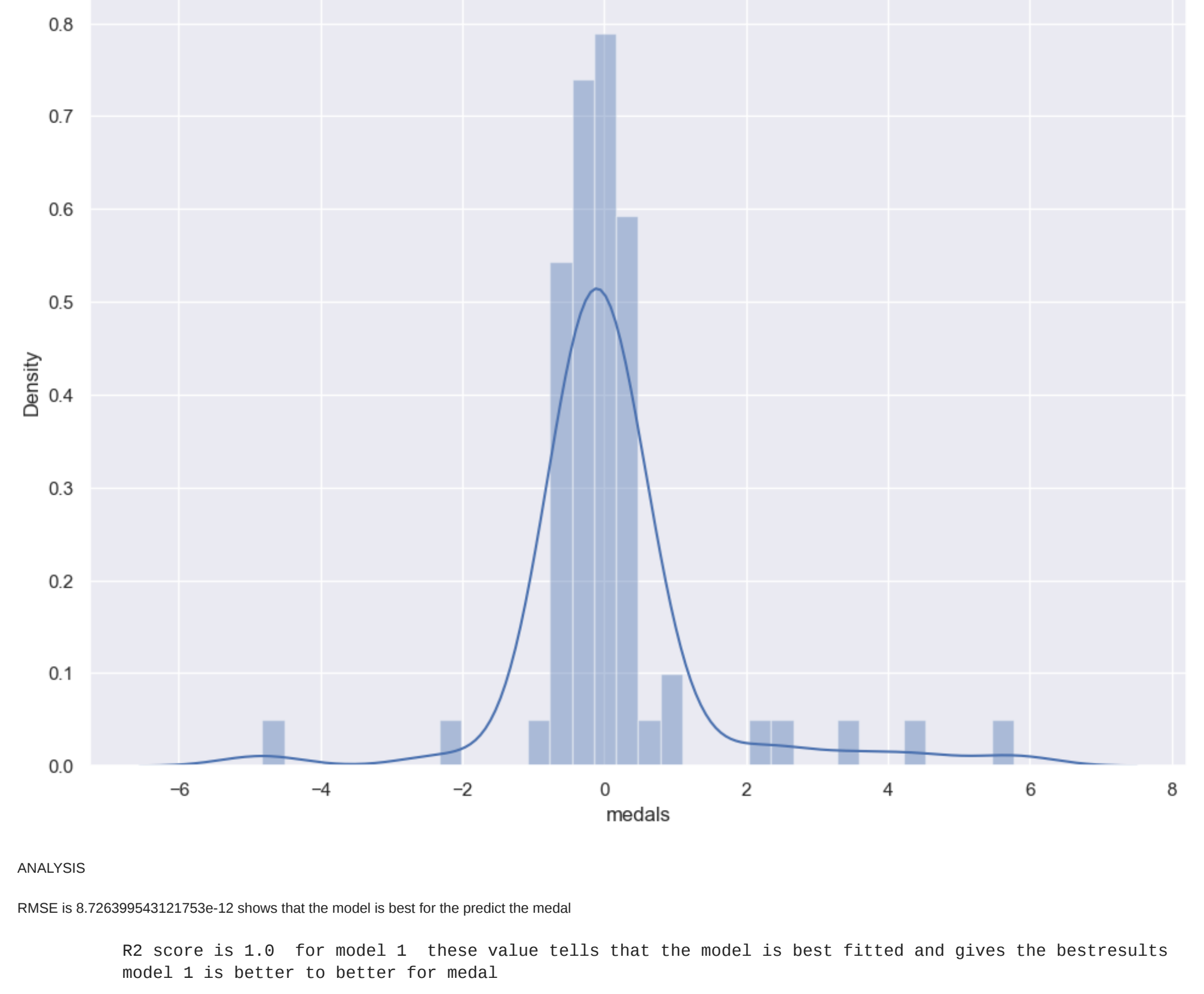
Out[270]:
```
<Axes: xlabel='medals', ylabel='Density'>
```



In [271…
```python
import seaborn as sns

sns.distplot(y_test-prediction_lasso)
```

Out[271]:
```
<Axes: xlabel='medals', ylabel='Density'>
```



ANALYSIS

RMSE is 8.726399543121753e-12 shows that the model is best for the predict the medal

```
R2 score is 1.0  for model 1  these value tells that the model is best fitted and gives the bestresults
model 1 is better to better for medal

the mean MAE value is 0 so the X1 model is best model to get uot put result for getting medal

  comparision the values of regreessions
Linear: -0.0, 0.0
Lasso: -9.78, 13.63
Ridge: -0.0, 0.0



mini_max showing that better accuracy value for 0.6004962779156328  function is showing
 standardized
```

model 2 also showing RMSE is 31.949238300307293

```
R2 score is 0.1759005375080599    is will be provides good results but not than model 1
```

In [ ]: