2306 AML128
BHASKARBABU ALAM

```python
import pandas as pd
import numpy as np
from sklearn import model_selection
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.linear_model import ElasticNet
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from math import sqrt

df = pd.read_csv('teams.csv')
print(df.shape)
df.describe()
```

```
(2014, 9)
```

	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

```
target_column = ['medals']
rem = ['team']
predictors = list(set(list(df.columns))-set(target_column)-set(rem))
#df[predictors] = df[predictors]/df[predictors].max()
df.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

```
X = df[predictors].values
y = df[target_column].values

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.30, random_state=40)
print(X_train.shape); print(X_test.shape)
```

```
(1409, 7)
(605, 7)
```

LINEAR REGRESSION

```
lr = LinearRegression()
lr.fit(X_train, y_train)
```

```
LinearRegression()
```

```
pred_train_lr= lr.predict(X_train)
print(np.sqrt(mean_squared_error(y_train,pred_train_lr)))
print(r2_score(y_train, pred_train_lr))

pred_test_lr= lr.predict(X_test)
print(np.sqrt(mean_squared_error(y_test,pred_test_lr)))
print(r2_score(y_test, pred_test_lr))
```

```
11.320950886277796
0.8976388322096625
12.465903124138876
0.8161419844397793
```

RIDGE REGRESSION

```
rr = Ridge(alpha=0.01)
rr.fit(X_train, y_train)
pred_train_rr= rr.predict(X_train)
print(np.sqrt(mean_squared_error(y_train,pred_train_rr)))
print(r2_score(y_train, pred_train_rr))

pred_test_rr= rr.predict(X_test)
print(np.sqrt(mean_squared_error(y_test,pred_test_rr)))
print(r2_score(y_test, pred_test_rr))
```

```
11.320950886277803
0.8976388322096623
12.465902934437699
0.8161419900355362
```

LASSO REGRESSION

```
model_lasso = Lasso(alpha=0.01)
model_lasso.fit(X_train, y_train)
pred_train_lasso= model_lasso.predict(X_train)
print(np.sqrt(mean_squared_error(y_train,pred_train_lasso)))
print(r2_score(y_train, pred_train_lasso))

pred_test_lasso= model_lasso.predict(X_test)
```

```
print(np.sqrt(mean_squared_error(y_test,pred_test_lasso)))
print(r2_score(y_test, pred_test_lasso))
```

```
11.320951816873057
0.8976388153812489
12.465391359015797
0.8161570800469944
```

Linear Regression Model: Test set RMSE of 12.465903124138876 and R-square of 81.61419844397793 percent.

Ridge Regression Model: Test set RMSE of 12.465902934437699 and R-square of 81.61419900355362 percent.

Lasso Regression Model: Test set RMSE of 12.465391359015797 and R-square of 81.61570800469944 percent.

Linear regression model has sligtly higher RSME value for the test data, and hence can be concluded as a less preferred method for this particular data set.

The perfect result would be an RMSE value of zero and R-squared value of 1, but that's almost impossible in real datasets

There are other iterations that can be done to improve model performance. We have assigned the value of alpha to be 0.01, but this can be altered by hyper parameter tuning to arrive at the optimal alpha value. Cross-validation can also be tried along with feature selection techniques