In [1]:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split

import warnings;
warnings.simplefilter('ignore')
```

In [2]:

```
df = pd.read_csv('teams.csv')
df.head()
```

Out[2]:

	team	year	athletes	events	age	height	weight	prev_medals	medals
0	AFG	1964	8	8	22.0	161.0	64.2	0.0	0
1	AFG	1968	5	5	23.2	170.2	70.0	0.0	0
2	AFG	1972	8	8	29.0	168.3	63.8	0.0	0
3	AFG	1980	11	11	23.6	168.4	63.2	0.0	0
4	AFG	2004	5	5	18.6	170.8	64.8	0.0	0

In [3]:

```
df.isna().sum()
```

Out[3]:

```
team           0
year           0
athletes       0
events         0
age            0
height         0
weight         0
prev_medals    0
medals         0
dtype: int64
```

In [4]:

```
train, test = train_test_split(df, test_size=0.3, random_state=1)
```

In [5]:

```
X=df.drop(['team','year','medals'],axis = 1)
y=df['medals']
```

In [6]:

```
X
```

Out[6]:

	athletes	events	age	height	weight	prev_medals
0	8	8	22.0	161.0	64.2	0.0
1	5	5	23.2	170.2	70.0	0.0
2	8	8	29.0	168.3	63.8	0.0
3	11	11	23.6	168.4	63.2	0.0
4	5	5	18.6	170.8	64.8	0.0
...	...	...	...	...	...	...
2009	26	19	25.0	179.0	71.1	0.0
2010	14	11	25.1	177.8	70.5	0.0
2011	16	15	26.1	171.9	63.7	3.0
2012	9	8	27.3	174.4	65.2	4.0
2013	31	13	27.5	167.8	62.2	0.0

2014 rows × 6 columns

In [7]:

```
y
```

Out[7]:

```
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 [8]:

```
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)
```

In [9]:

```
X_train.head()
```

Out[9]:

	athletes	events	age	height	weight	prev_medals
376	730	257	23.9	176.0	69.9	94.0
1253	8	8	28.1	169.7	75.8	0.0
1279	22	17	20.0	184.8	78.5	0.0
1173	42	29	23.3	169.3	66.8	0.0
1286	7	7	26.0	175.3	78.7	0.0

In [10]:

```
X_test.head()
```

Out[10]:

	athletes	events	age	height	weight	prev_medals
1911	23	11	22.0	186.5	81.5	0.0
1686	3	3	27.3	168.0	57.3	0.0
1179	2	2	26.5	171.0	60.0	0.0
953	4	4	24.0	165.0	69.0	0.0
1364	6	6	20.8	179.2	70.3	0.0

In [11]:

```
y_train.head()
```

Out[11]:

```
376     184
1253      0
1279      0
1173      1
1286      0
Name: medals, dtype: int64
```

In [12]:

```
y_test.head()
```

Out[12]:

```
1911    0
1686    0
1179    0
953     0
1364    1
Name: medals, dtype: int64
```

Linear Regression

In [13]:

```
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

In [14]:

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

Out[14]:

```
▾ LinearRegression

LinearRegression()
```

In [15]:

```
y_pred = model.predict(X_test)
```

In [16]:

```
model.score(X_train, y_train)
```

Out[16]:

```
0.8780900000469467
```

In [17]:

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

Out[17]:

```
0.903162022187423
```

In [18]:

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

mse = cross_val_score(linear_regressor, X, y, cv=5, scoring='neg_mean_squared_error')
rmse = np.sqrt(mse)
r2 =  r2_score(y_test, y_pred)

mean_mse = np.mean(mse)
mean_rmse = np.mean(rmse)
mean_r2 = np.mean(r2)

print("Mean Squared Error: ",mean_mse)
print("Root Mean Squared Error:",mean_rmse)
print("R-squared (R²):",mean_r2)
```

```
Mean Squared Error:  -145.42031462787676
Root Mean Squared Error: nan
R-squared (R²): 0.903162022187423
```

Ridge Regression

In [19]:

```
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_squared_error, r2_score

ridge_regressor = Ridge(alpha=1.0)
ridge_regressor.fit(X_train, y_train)
```

Out[19]:

```
▾ Ridge
Ridge()
```

In [20]:

```
y_pred = ridge_regressor.predict(X_test)

mse = cross_val_score(ridge_regressor, X, y, cv=5, scoring='neg_mean_squared_error')
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)


mean_mse = np.mean(mse)
mean_rmse = np.mean(rmse)
mean_r2 = np.mean(r2)

print("Mean Squared Error (MSE):", mean_mse)
print("Root Mean Squared Error (RMSE):", mean_rmse)
print("R-squared (R²):", mean_r2)
```

```
Mean Squared Error (MSE): -145.42048569294258
Root Mean Squared Error (RMSE): nan
R-squared (R²): 0.9031620494510695
```

In [21]:

```
ridge_regressor.score(X_train, y_train)
```

Out[21]:

```
0.8780900000459477
```

In [22]:

```
ridge_regressor.score(X_test, y_test)
```

Out[22]:

```
0.9031620494510695
```

Lasso Regression

In [23]:

```
from sklearn.linear_model import Lasso
from sklearn.metrics import mean_squared_error, r2_score

lasso_regressor = Lasso(alpha = 1.0)

lasso_regressor.fit(X_train, y_train)
```

Out[23]:

```
▾ Lasso
Lasso()
```

In [24]:

```
y_pred = lasso_regressor.predict(X_test)

mse = cross_val_score(ridge_regressor, X, y, cv=5, scoring='neg_mean_squared_error')
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

mean_mse = np.mean(mse)
mean_rmse = np.mean(rmse)
mean_r2 = np.mean(r2)

print("Mean Squared Error (MSE):", mse)
print("Root Mean Squared Error (RMSE):", rmse)
print("R-squared (R²):", r2)
```

```
Mean Squared Error (MSE): [-101.55761296 -151.11313651  -80.73396609  -69.
21135878 -324.48635412]
Root Mean Squared Error (RMSE): [nan nan nan nan nan]
R-squared (R²): 0.9036413581449435
```

In [25]:

```
lasso_regressor.score(X_train, y_train)
```

Out[25]:

```
0.8780364628849183
```

In [26]:

```
lasso_regressor.score(X_test, y_test)
```

Out[26]:

```
0.9036413581449435
```

In [27]:

```
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

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(),
```

```
Linear: -158.19, 128.82
Lasso: -158.23, 129.35
Ridge: -158.19, 128.82
```