Supervised Machine learning Model

Data from historical Olympic games

Project steps:

1. Form a Hypothesis.
    - Does Mexico perform well in Olympic games?
2. Find and explore the data.
    - Use Olympic games dataset.
3. (If necessary) Reshape the data to predict your target.
    - Our target should be to predict how many medals does a country win in contrast with previous years.
4. Clean the data
    - Does the dataset need cleaning?
5. Pick an error metric.
    - How far off are we with predictions from our model and actual data? (mean absolute error)
6. Split your data
    - Usually 80% for training and 20% for test.
7. Train a model.
    - Choose an adecuate model (Linear Regression)

In [ ]:

```
#Import Libraries
import pandas as pd
import seaborn as sns
import numpy as np
from sklearn.linear_model import LinearRegression
```

In [2]:

```
#Load dataset
teams = pd.read_csv("teams.csv")
```

In [3]:

```
teams
```

Out[3]:

	team	country	year	events	athletes	age	height	weight	medals	prev_medals	prev_3_n
0	AFG	Afghanistan	1964	8	8	22.0	161.0	64.2	0	0.0	
1	AFG	Afghanistan	1968	5	5	23.2	170.2	70.0	0	0.0	
2	AFG	Afghanistan	1972	8	8	29.0	168.3	63.8	0	0.0	
3	AFG	Afghanistan	1980	11	11	23.6	168.4	63.2	0	0.0	
4	AFG	Afghanistan	2004	5	5	18.6	170.8	64.8	0	0.0	
...	...	...	...	...	...	...	...	...	...	...	
2139	ZIM	Zimbabwe	2000	19	26	25.0	179.0	71.1	0	0.0	
2140	ZIM	Zimbabwe	2004	11	14	25.1	177.8	70.5	3	0.0	
2141	ZIM	Zimbabwe	2008	15	16	26.1	171.9	63.7	4	3.0	
2142	ZIM	Zimbabwe	2012	8	9	27.3	174.4	65.2	0	4.0	
2143	ZIM	Zimbabwe	2016	13	31	27.5	167.8	62.2	0	0.0	

2144 rows × 11 columns

In [4]:

```
#Remove extra-columns
teams = teams[["team", "country", "year","athletes", "age", "prev_medals", "medals"]]
```

In [5]:

```
teams
```

Out[5]:

	team	country	year	athletes	age	prev_medals	medals
0	AFG	Afghanistan	1964	8	22.0	0.0	0
1	AFG	Afghanistan	1968	5	23.2	0.0	0
2	AFG	Afghanistan	1972	8	29.0	0.0	0
3	AFG	Afghanistan	1980	11	23.6	0.0	0
4	AFG	Afghanistan	2004	5	18.6	0.0	0
...	...	...	...	...	...	...	...
2139	ZIM	Zimbabwe	2000	26	25.0	0.0	0
2140	ZIM	Zimbabwe	2004	14	25.1	0.0	3
2141	ZIM	Zimbabwe	2008	16	26.1	3.0	4
2142	ZIM	Zimbabwe	2012	9	27.3	4.0	0
2143	ZIM	Zimbabwe	2016	31	27.5	0.0	0

2144 rows × 7 columns

In [55]:

```
#List of countries in dataset
teams["country"].unique()
```

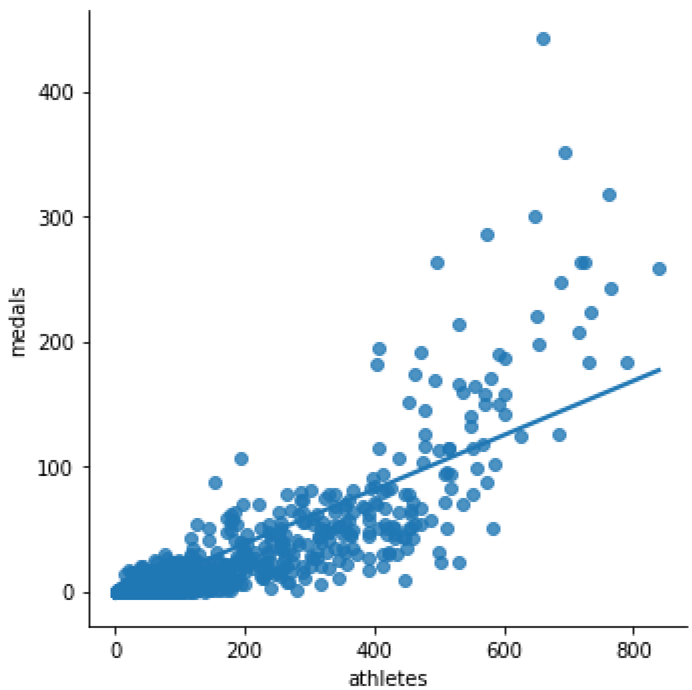
Out[55]:
```
array(['Afghanistan', 'Netherlands Antilles', 'Albania', 'Algeria',
       'Andorra', 'Angola', 'Antigua and Barbuda', 'Argentina', 'Armenia',
       'Aruba', 'American Samoa', 'Australia', 'Australia-2',
       'Australia-1', 'Austria', 'Azerbaijan', 'John B', 'Bahamas',
       'Bangladesh', 'Barbados', 'Burundi', 'Belgium', 'Benin',
       'Oleander XII', 'Bermuda', 'Bhutan', 'Bosnia and Herzegovina',
       'Belize', 'Belarus', 'Bolivia', 'Botswana', 'Brazil', 'Bahrain',
       'Brunei', 'Bulgaria', 'Burkina Faso', 'Central African Republic',
       'Cambodia', 'Canada', 'Canada-2', 'Cayman Islands',
       'Congo (Brazzaville)', 'Chad', 'Chile', 'China', "Cote d'Ivoire",
       'Cameroon', 'Congo (Kinshasa)', 'Cook Islands', 'Colombia',
       'Comoros', 'Cape Verde', 'Costa Rica', 'Croatia', 'Cuba', 'Cyprus',
       'Czech Republic', 'Denmark', 'Digby', 'Djibouti', 'Dominica',
       'Dominican Republic', 'Ecuador', 'Egypt', 'Eritrea', 'El Salvador',
       'Spain', 'Estonia', 'Ethiopia', 'Fiji', 'Finland', 'France',
       'West Germany', 'Federated States of Micronesia', 'Gabon',
       'Gambia', 'Great Britain', 'Guinea Bissau', 'East Germany',
       'Georgia', 'Equatorial Guinea', 'Germany', 'Ghana', 'Proteus II',
       'Greece', 'Grenada', 'Guatemala', 'Guinea', 'Guam', 'Guyana',
       'Haiti', 'Hong Kong', 'Honduras', 'Hungary', 'Indonesia', 'India',
       'Individual Olympic Athletes', 'Iran', 'Ireland', 'Iraq',
       'Iceland', 'Israel', 'United States Virgin Islands', 'Italy',
       'British Virgin Islands', 'Miss Nippon IV', 'Jamaica', 'Jordan',
       'Japan', 'Japan-1', 'Kazakhstan', 'Kenya', 'Kyrgyzstan',
       'Kiribati', 'South Korea', 'Saudi Arabia', 'Kuwait', 'Laos',
       'Latvia', 'Libya', 'Liberia', 'Saint Lucia', 'Lesotho', 'Lebanon',
       'Liechtenstein', 'Lithuania', 'Luxembourg', 'Madagascar',
       'Morocco', 'Malaysia', 'Malaysia-1', 'Malawi', 'Moldova',
       'Maldives', 'Mexico', 'Mongolia', 'Marshall Islands', 'Macedonia',
       'Mali', 'Malta', 'Montenegro', 'Monaco', 'Mozambique', 'Mauritius',
       'Mauritania', 'Myanmar', 'Namibia', 'Nicaragua', 'Netherlands',
       'Nepal', 'Nigeria', 'Niger', 'Norway', 'Nauru', 'New Zealand',
       'Oman', 'Pakistan', 'Panama', 'Paraguay', 'Peru', 'Kalayaan',
       'Philippines', 'Palestine', 'Palau', 'Papua New Guinea', 'Poland',
       'Portugal', 'North Korea', 'North Korea-1', 'Puerto Rico', 'Qatar',
       'Romania', 'South Africa', 'Russia', 'Russia-2', 'Rwanda', 'Samoa',
       'Serbia and Montenegro', 'Senegal', 'Seychelles', 'Singapore',
       'Singapore-1', 'Saint Kitts and Nevis', 'Sierra Leone', 'Slovenia',
       'San Marino', 'Solomon Islands', 'Somalia', 'Serbia', 'Sri Lanka',
       'Sao Tome and Principe', 'Sudan', 'Switzerland', 'Switzerland-1',
       'Suriname', 'Slovakia', 'Sweden', 'Swaziland', 'Syria', 'Tanzania',
       'Czechoslovakia', 'Tonga', 'Thailand', 'Tajikistan',
       'Turkmenistan', 'Timor Leste', 'Togo', 'Chinese Taipei',
       'Chinese Taipei-1', 'Trinidad and Tobago', 'Tunisia', 'Turkey',
       'Tuvalu', 'United Arab Emirates', 'Uganda', 'Ukraine',
       'Soviet Union', 'Uruguay', 'United States', 'Uzbekistan',
       'Vanuatu', 'Venezuela', 'Vietnam',
       'Saint Vincent and the Grenadines', 'South Vietnam', 'North Yemen',
       'Yemen', 'Yugoslavia', 'Zambia', 'Zimbabwe'], dtype=object)
```

In [7]:
```
#which columns are correlated so that we can use them
#correlation for athletes and previous medals is high
#correlation for year and age is low
teams.corr()["medals"]
```

Out[7]:
```
year          -0.021603
athletes       0.840817
age            0.025096
prev_medals    0.920048
medals         1.000000
Name: medals, dtype: float64
```

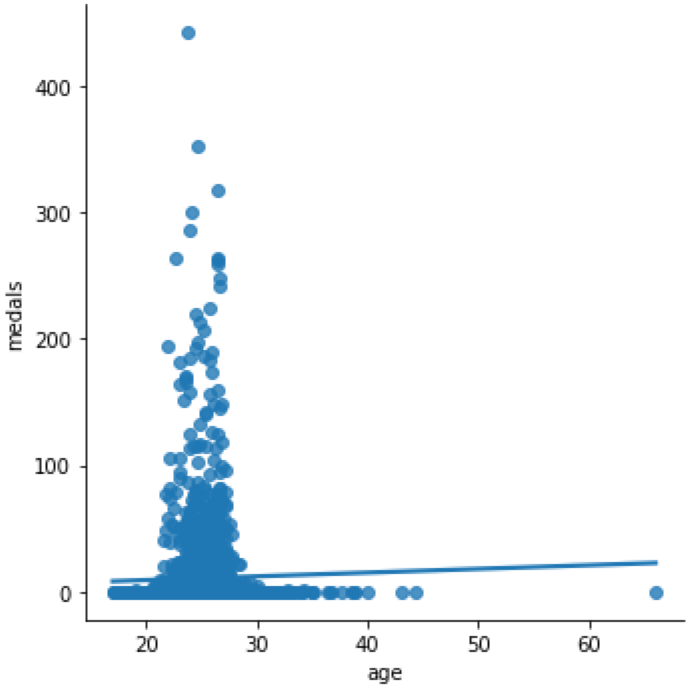
In [9]:
```
#Plotting athletes vs medals
#This plot tells us there is a higher chance of getting a medal as the number of athle
sns.lmplot(x="athletes", y="medals", data=teams, fit_reg=True, ci=None)
```

Out[9]:
```
<seaborn.axisgrid.FacetGrid at 0x1c09ddc25c0>
```



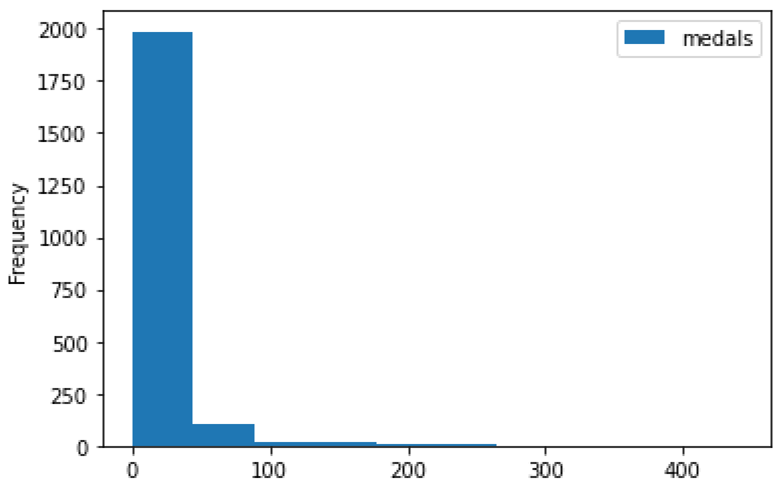
In [10]:
```
#Plotting age vs medals
# There might be a correlation between athletes being age(20-30) and winning a medal.
sns.lmplot(x="age", y="medals", data=teams, fit_reg=True, ci=None)
```

Out[10]:
```
<seaborn.axisgrid.FacetGrid at 0x1c09de81930>
```



In [11]:
```
#This histogram shows that only few countries earn a lot of medals.
teams.plot.hist(y="medals")
```

Out[11]:
```
<AxesSubplot:ylabel='Frequency'>
```



In [12]:
```
#Finding missing values
teams[teams.isnull().any(axis=1)]
```

Out[12]:

	team	country	year	athletes	age	prev_medals	medals
19	ALB	Albania	1992	9	25.3	NaN	0
26	ALG	Algeria	1964	7	26.0	NaN	0
39	AND	Andorra	1976	3	28.3	NaN	0
50	ANG	Angola	1980	17	17.4	NaN	0
59	ANT	Antigua and Barbuda	1976	17	23.2	NaN	0
...	...	...	...	...	...	...	...
2092	VIN	Saint Vincent and the Grenadines	1988	6	20.5	NaN	0
2103	YAR	North Yemen	1984	3	27.7	NaN	0
2105	YEM	Yemen	1992	8	19.6	NaN	0
2112	YMD	South Yemen	1988	5	23.6	NaN	0
2120	ZAM	Zambia	1964	15	21.7	NaN	0

130 rows × 7 columns

In [13]:
```
#Eliminate rows with missing values
teams = teams.dropna()
```

In [14]:
```
teams
```

Out[14]:

	team	country	year	athletes	age	prev_medals	medals
0	AFG	Afghanistan	1964	8	22.0	0.0	0
1	AFG	Afghanistan	1968	5	23.2	0.0	0
2	AFG	Afghanistan	1972	8	29.0	0.0	0
3	AFG	Afghanistan	1980	11	23.6	0.0	0
4	AFG	Afghanistan	2004	5	18.6	0.0	0
...	...	...	...	...	...	...	...
2139	ZIM	Zimbabwe	2000	26	25.0	0.0	0
2140	ZIM	Zimbabwe	2004	14	25.1	0.0	3
2141	ZIM	Zimbabwe	2008	16	26.1	3.0	4
2142	ZIM	Zimbabwe	2012	9	27.3	4.0	0
2143	ZIM	Zimbabwe	2016	31	27.5	0.0	0

2014 rows × 7 columns

In [16]:
```
#Split dataset
train = teams[teams["year"] < 2012].copy()
test = teams[teams["year"] >= 2012].copy()
```

In [17]:
```
# 80% train
train.shape
```
Out[17]:
```
(1609, 7)
```
In [18]:
```
# 20% test
test.shape
```
Out[18]:
```
(405, 7)
```
In [20]:
```
# Linear regression function
reg = LinearRegression()
```
In [21]:
```
predictors = ["athletes", "prev_medals"]
target = "medals"
```
In [22]:
```
# fit our linear regression model
reg.fit(train[predictors], train["medals"])
```
Out[22]:
```
▾ LinearRegression
LinearRegression()
```
In [23]:
```
# Create predictions
predictions = reg.predict(test[predictors])
```
In [25]:
```
test["predictions"] = predictions
```
In [27]:
```
# values cannot be negative or decimals
test.loc[test["predictions"] < 0, "predictions"] = 0
```
In [29]:
```
test["predictions"] = test["predictions"].round()
```
In [30]:
```
test
```

Out[30]:

	team	country	year	athletes	age	prev_medals	medals	predictions
6	AFG	Afghanistan	2012	6	24.8	1.0	1	0.0
7	AFG	Afghanistan	2016	3	24.7	1.0	0	0.0
24	ALB	Albania	2012	10	25.7	0.0	0	0.0
25	ALB	Albania	2016	6	23.7	0.0	0	0.0
37	ALG	Algeria	2012	39	24.8	2.0	1	2.0
...	...	...	...	...	...	...	...	...
2111	YEM	Yemen	2016	3	19.3	0.0	0	0.0
2131	ZAM	Zambia	2012	7	22.6	0.0	0	0.0
2132	ZAM	Zambia	2016	7	24.1	0.0	0	0.0
2142	ZIM	Zimbabwe	2012	9	27.3	4.0	0	2.0
2143	ZIM	Zimbabwe	2016	31	27.5	0.0	0	0.0

405 rows × 8 columns

In [32]:

```
from sklearn.metrics import mean_absolute_error

error = mean_absolute_error(test["medals"], test["predictions"])
```

In [33]:

```
# margin of error
error
```

Out[33]:

```
3.2987654320987656
```

In [34]:

```
teams.describe()["medals"]
```

Out[34]:

```
count    2014.000000
mean       10.990070
std        33.627528
min         0.000000
25%         0.000000
50%         0.000000
75%         5.000000
max       442.000000
Name: medals, dtype: float64
```

In [35]:

```
# looking at predicions with a high medal standing country
test[test["team"] == "USA"]
```

Out[35]:

	team	country	year	athletes	age	prev_medals	medals	predictions
2053	USA	United States	2012	689	26.7	317.0	248	285.0
2054	USA	United States	2016	719	26.4	248.0	264	236.0

In [57]:

```
# looking at prediction with a low medal standing country
test[test["team"] == "MEX"]
```

Out[57]:

	team	country	year	athletes	age	prev_medals	medals	predictions
1285	MEX	Mexico	2012	119	25.2	4.0	24	9.0
1286	MEX	Mexico	2016	139	25.4	24.0	5	26.0

In [37]:

```
# we want to show the relation of medals vs predictions better
# This will show us errors by country
errors = (test["medals"] - test["predictions"]).abs()
```

In [38]:

```
errors
```

Out[38]:

```
6       1.0
7       0.0
24      0.0
25      0.0
37      1.0
       ...
2111    0.0
2131    0.0
2132    0.0
2142    2.0
2143    0.0
Length: 405, dtype: float64
```

In [39]:

```
#Group errors by team
error_by_team = errors.groupby(test["team"]).mean()
```

In [40]:

```
# How many medals off are we
error_by_team
```

Out[40]:

```
team
AFG    0.5
ALB    0.0
ALG    1.5
AND    0.0
ANG    0.0
      ...
VIE    1.0
VIN    0.0
YEM    0.0
ZAM    0.0
ZIM    1.0
Length: 204, dtype: float64
```

In [44]:

```
# How many medals on average by team
medals_by_team = test["medals"].groupby(test["team"]).mean()
```

In [47]:

```
error_ratio = error_by_team / medals_by_team
```

In [49]:

```
# Getting rid of missing values
error_ratio[~pd.isnull(error_ratio)]
```

Out[49]:
```
team
AFG    1.000000
ALG    1.000000
ARG    0.853659
ARM    0.428571
AUS    0.367347
         ...
USA    0.126953
UZB    0.625000
VEN    1.750000
VIE    1.000000
ZIM         inf
Length: 102, dtype: float64
```

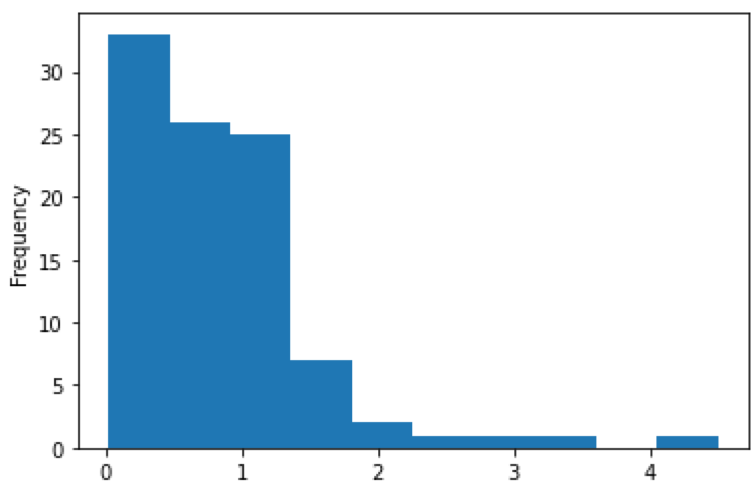
In [51]:
```
# Getting rid of infinite values
error_ratio = error_ratio[np.isfinite(error_ratio)]
```

In [52]:
```
# Error ratio table
error_ratio
```

Out[52]:
```
team
AFG    1.000000
ALG    1.000000
ARG    0.853659
ARM    0.428571
AUS    0.367347
         ...
UKR    0.951220
USA    0.126953
UZB    0.625000
VEN    1.750000
VIE    1.000000
Length: 97, dtype: float64
```

In [53]:
```
# This shows that for some countries the predictions were far off
error_ratio.plot.hist()
```

Out[53]:
```
<AxesSubplot:ylabel='Frequency'>
```



In [54]:
```
# We can see that for countries that tend to win medals in every olympic games the err
# and for countries that we barely see in olympic games the error ratio is very high
error_ratio.sort_values()
```

Out[54]:
```
team
FRA    0.022472
CAN    0.048387
NZL    0.063492
RUS    0.082353
ITA    0.121429
         ...
MAR    2.000000
EGY    2.400000
HKG    3.000000
POR    3.333333
AUT    4.500000
Length: 97, dtype: float64
```

In [58]:
```
# Try randomforestclassifier
from sklearn.ensemble import RandomForestClassifier

forest = RandomForestClassifier()
forest.fit(train[predictors], train["medals"])
```

Out[58]:
```
RandomForestClassifier
RandomForestClassifier()
```

In [59]:
```
forest_predictions = forest.predict(test[predictors])
```

In [60]:
```
test["predictions"] = forest_predictions
```

In [61]:
```
test
```

Out[61]:

	team	country	year	athletes	age	prev_medals	medals	predictions
6	AFG	Afghanistan	2012	6	24.8	1.0	1	0
7	AFG	Afghanistan	2016	3	24.7	1.0	0	0
24	ALB	Albania	2012	10	25.7	0.0	0	0
25	ALB	Albania	2016	6	23.7	0.0	0	0
37	ALG	Algeria	2012	39	24.8	2.0	1	1
...	...	...	...	...	...	...	...	...
2111	YEM	Yemen	2016	3	19.3	0.0	0	0
2131	ZAM	Zambia	2012	7	22.6	0.0	0	0
2132	ZAM	Zambia	2016	7	24.1	0.0	0	0
2142	ZIM	Zimbabwe	2012	9	27.3	4.0	0	2
2143	ZIM	Zimbabwe	2016	31	27.5	0.0	0	0

405 rows × 8 columns

In [62]:
```
forest_error = mean_absolute_error(test["medals"], test["predictions"])
```

In [63]:

```
forest_error
```

Out[63]:

```
4.706172839506173
```

Conclusions

1.Error seems to be greater than a linear regression. 2.Would be wise to use other predictors and try to find another model that fits better.