Machine learning Project

Step 1: Choosing a hypothesis

Step 2: Looking for dataset

Step 3 : Reshape data

Step 4: Clean data - handling missing values

Step 5: Finding the error metric - to evaluate the performance

Step 6: Split the data - into training and testing data

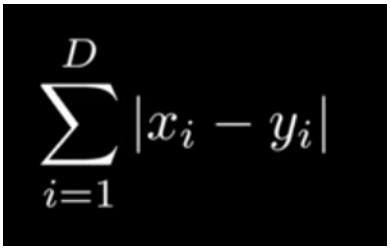
Step 7: Train the model

What are we predicting?

⇒ How many medals a country should have earned in a given Olympics

What error metric will be used?

⇒ Mean absolute error



Here error column is added = Actual medals - predicted values

While model are you using to do the training?

⇒ Linear regression

⇒ Univariate linear regression. Only one variable. (medal in previous Olympics to predict the medals a country will get in this Olympics)

⇒ In our actual model we are going to use bi-variate

$$Y = a_1x_1 + a_2x_2 + B$$

Upon checking for the correlation between the ‘medals’ column and the other columns we find the following:

```
[25] corr = teams.drop(["team","country"],axis=1).corr()["medals"]
     print(corr)

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

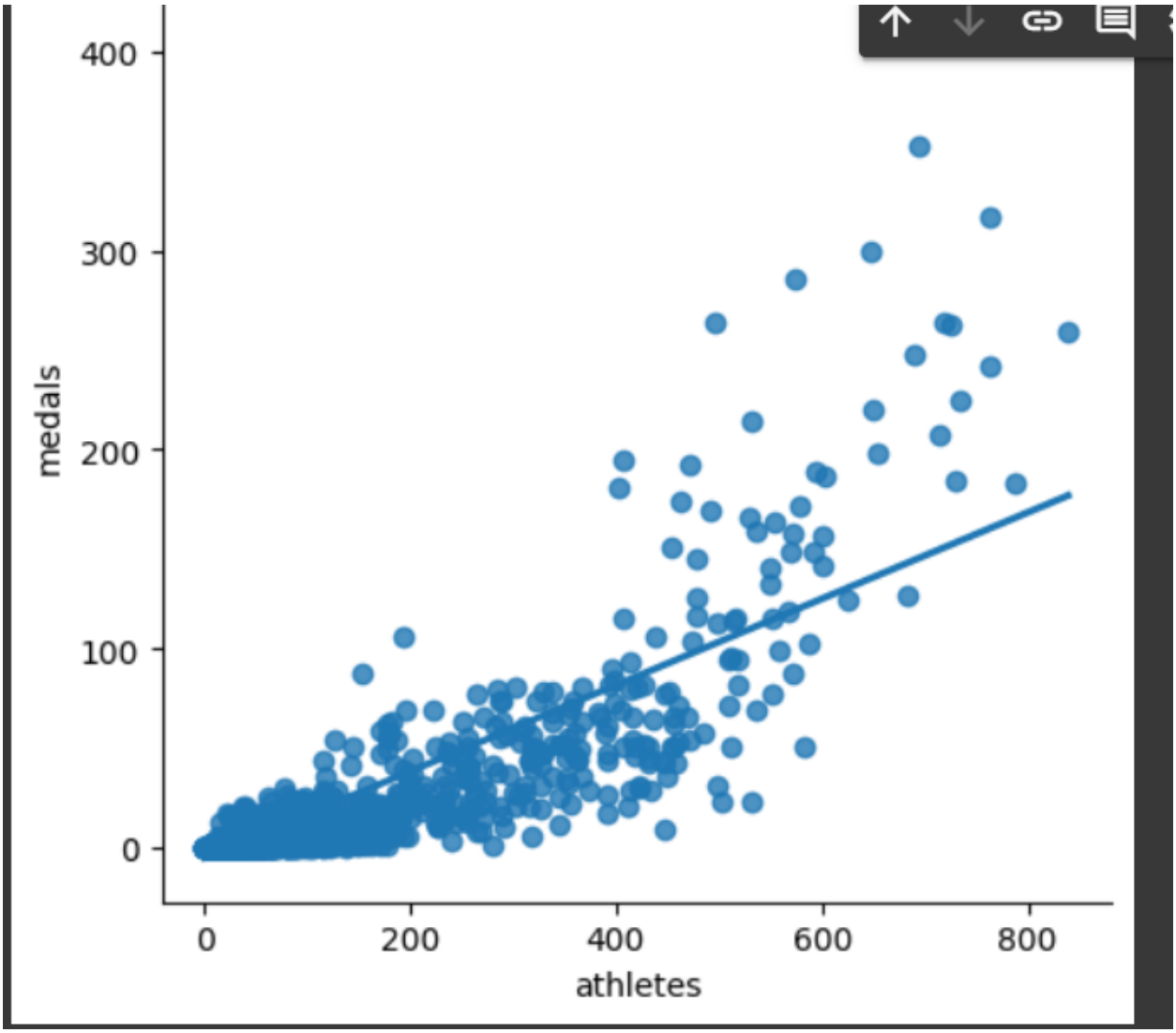
First we consider the correlation between athletes and the medals column because the correlation seems to be on the higher side.

We use sns python graphing library.

```
sns.lmplot(x="athletes", y="medals", data=teams, fit_reg=True, ci=None)
```

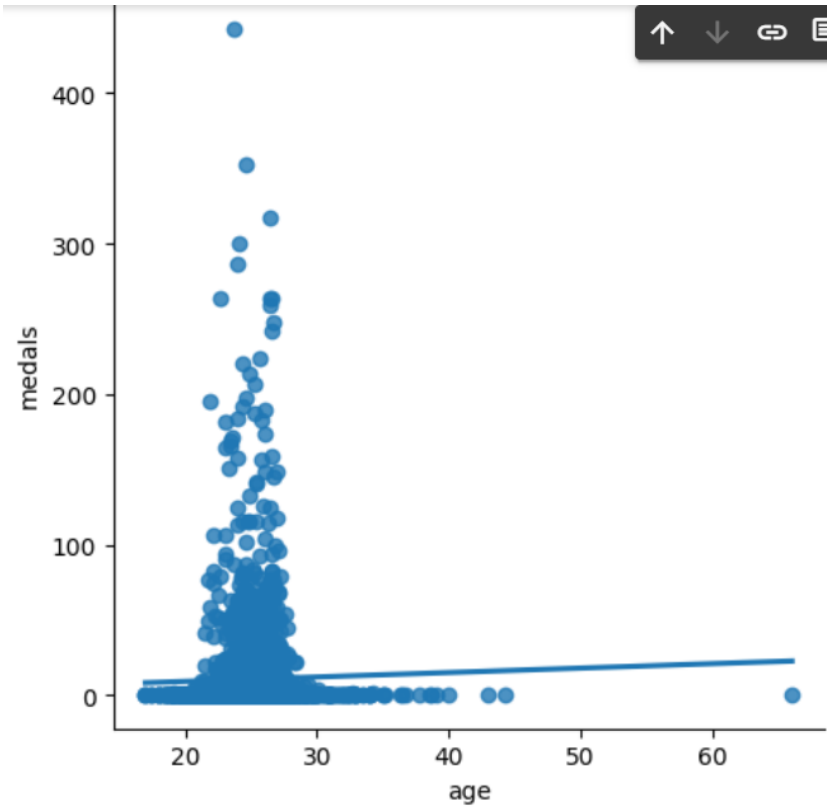
Here the it will fit a regression

Here is the output:



We see that there is a linear relationship between the athletes and the medals earned. This makes sense because the more the number of athletes that joined, the more are the chances of winning a medal for a country.

Next we look into the column where there isn't a lot of correlation, i.e, age

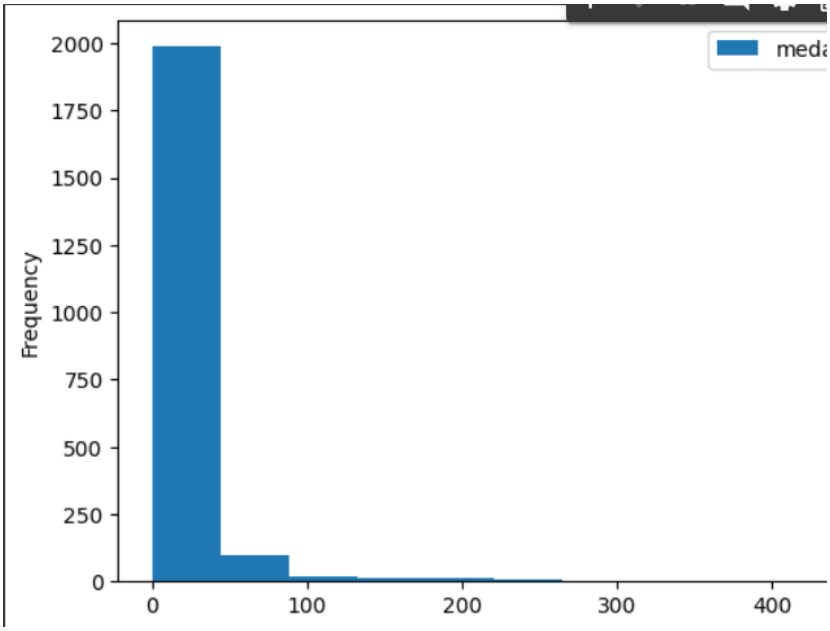


From the output its clear that there isn't a linear relationship between the age of the athletes and the medals won.

Looking into a histogram to see how many countries got how many medals.

Code: `teams.plot.hist(y="medals")`

Output-



Thid output id imbalanced.

Looking for Null values:

```
teams[teams.isnull().any(axis=1)]
```

	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

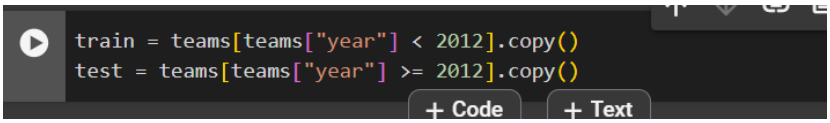
To drop Null values:

```
teams = teams.dropna()
```

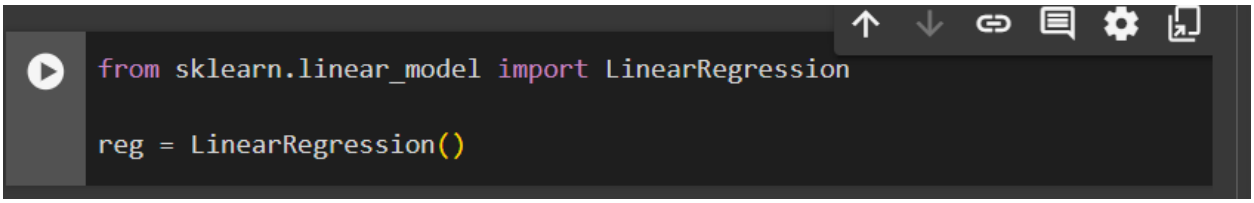
Why do we have null values?

⇒ Because there might have been a chance that a team did not participate in the previous year.

Next we need to make sure that the we do not use the future's data to predict the past.
Hence, we did not do a random split. Instead we chose a threshold year and split the data into train and test accordingly.



We are getting our linear regression model from sklearn library:

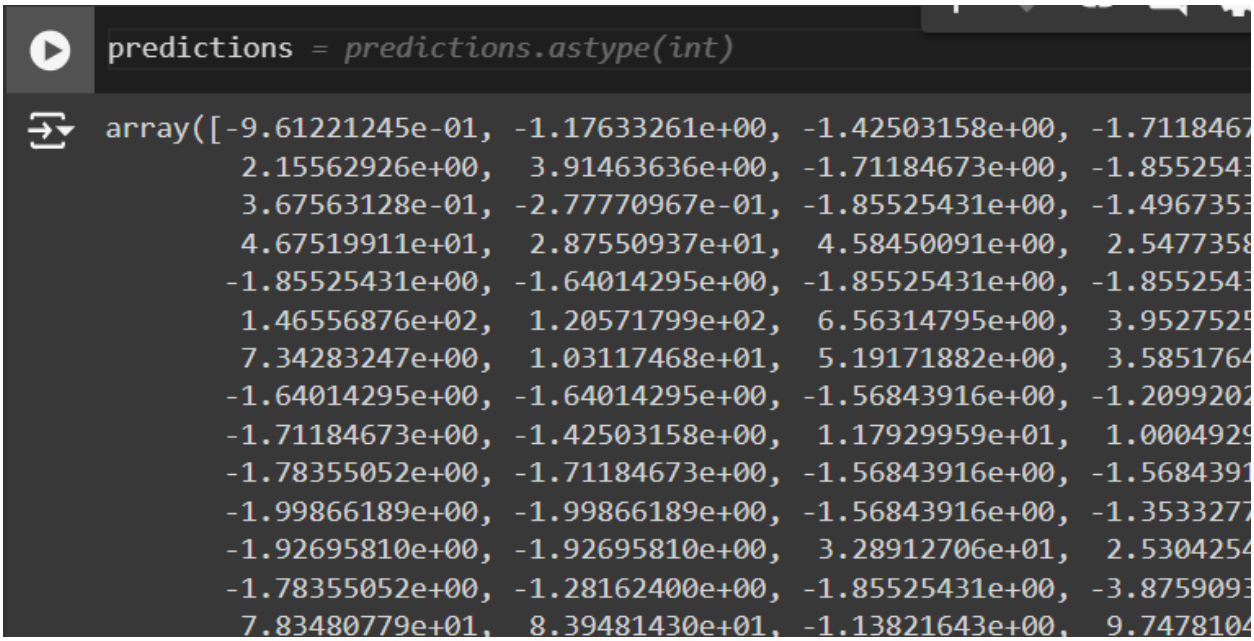


This is followed by the below codes for continuing with the prediction:

```
predictors = ["athletes","prev_medals"]
target = "medals"
```

```
reg.fit(train[predictors], train["medals"])
```

```
predictors = reg.predict(test[predictors])
```



Well the predictions need to be a whole number because we are predicting the number of medals gained.

```
test.loc[test["predictions"]< 0, "predictions"] = 0
```

We use this code to set the prediction values to 0 where the predicted values are less than 0.

Error:

Do a Sanity check where you need to make sure that your error must be less than the standard deviation.