```
#Importing Libraries

import numpy as np     #to do numerical computing
import pandas as pd    #we can data manipulate and analysis


df = pd.read_csv('/content/teams.csv')     #loading the datadet
df.head()                                  #this function will show first 5 rows from the dataset
```

	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

```
df.shape      #This function will show how many rows and columns in the dataset
```

```
(2014, 9)
```

```
df.columns    #This function will show all column names
```

```
Index(['team', 'year', 'athletes', 'events', 'age', 'height', 'weight',
       'prev_medals', 'medals'],
      dtype='object')
```

```
df.info()     #This function will show datatype in each column and no.of null values
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2014 entries, 0 to 2013
Data columns (total 9 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   team         2014 non-null   object
 1   year         2014 non-null   int64
 2   athletes     2014 non-null   int64
 3   events       2014 non-null   int64
 4   age          2014 non-null   float64
 5   height       2014 non-null   float64
 6   weight       2014 non-null   float64
 7   prev_medals  2014 non-null   float64
 8   medals       2014 non-null   int64
dtypes: float64(4), int64(4), object(1)
memory usage: 141.7+ KB
```

```
df.isna().sum()       #This function will show null values in each column
```

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

```
df['medals'].unique()       #THs function will show no.of unique values in medals column
```

```
array([  0,   1,   2,   3,   5,  13,  20,  49,  51,  19,  22,   6,   4,
        44,  23,  52,  35,  57, 132, 183, 157, 149, 114,  82,   8,   7,
        10,  18,  11,   9,  21,  15,  29,  12,  36,  14,  63,  48,  40,
        78,  59,  50,  26,  39,  90,  41,  17,  87,  31,  55,  69,  74,
       106,  79,  94, 184, 125, 113,  28,  24,  65,  62,  47,  25,  16,
        66,  42,  30,  70,  45,  27,  67,  53,  77,  96, 102, 158, 115,
        32,  71,  54,  81, 126, 145, 151, 195, 264, 181, 116, 198, 124,
       118,  99, 159,  56,  61,  43,  33,  37, 104,  68,  72,  93,  84,
        64,  73,  46,  58,  38, 187, 189, 142, 140,  34, 174, 192, 214,
       286, 442, 300, 169, 166, 171, 164, 352, 207, 224, 259, 242, 263,
       317, 248])
```

```
df.describe()       #This fuction will show count, mean, standard deviation, minimum value, 25th percentile, median, 75th percentile, max
```

	year	athletes	events	age	height	weight	prev_medals	medals
count	2014.000000	2014.000000	2014.000000	2014.000000	2014.000000	2014.000000	2014.000000	2014.000000
mean	1995.227408	76.329692	36.877855	24.812612	173.955164	69.328997	10.248759	10.990070
std	15.227727	129.799427	50.130877	2.758258	5.262469	7.494740	31.951920	33.627528
min	1964.000000	1.000000	1.000000	17.000000	151.000000	43.300000	0.000000	0.000000
25%	1984.000000	7.000000	6.000000	23.300000	170.600000	64.700000	0.000000	0.000000
50%	1996.000000	21.000000	14.000000	24.700000	174.400000	69.500000	0.000000	0.000000
75%	2008.000000	74.750000	47.000000	26.100000	177.800000	73.400000	4.000000	5.000000

```
X=df.drop(['team','year','medals'],axis = 1)     #Here we are splitting the dataset into independent and dependent features
y=df['medals']
```

X

	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

y

```
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
```

```
#Splitting the dataset to train and test
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)


#Applying standard scaler -- it will normalise the features of dataset
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)


#importing Linear Regression Algorithm to understand and predict
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
model = lr.fit(X_train, y_train)    #this function is used to train the dataset
y_pred = model.predict(X_test)      #predict model


#importing metrics to know the performance of model
from sklearn.metrics import r2_score, mean_squared_error
mse = mean_squared_error(y_test,y_pred)
print(mse)
```

```
132.55472031550494
```

```
r2 = r2_score(y_test, y_pred)
print(r2)
```

```
0.8738705405726803
```

```
#creating ANN model
import keras
from keras.models import Sequential
from keras.layers import Dense #sparse or dense
from keras.layers import LeakyReLU,PReLU,ELU # activation functions
from tensorflow.keras.layers import Dropout


# intialize the empty artifical neural network without inputs and outputs
model = Sequential()
model.add(Dense(units = 20, kernel_initializer = 'he_uniform', activation = 'relu', input_dim = 6))    #first layer with 6 inputs
model.add(Dropout(0.5))  #to drop the unnecessary neurons
model.add(Dense(units = 10,kernel_initializer = 'he_uniform', activation = 'relu'))                     #hidden layer 1
model.add(Dense(units = 5, kernel_initializer= 'he_uniform', activation = 'relu'))                      #hidden layer 2
model.add(Dense(units = 1, kernel_initializer = 'glorot_uniform', activation = 'softmax'))              #last later it will give the outpu


#compling the ANN
model.compile(optimizer = 'Adam' , loss = "mean_squared_error" , metrics = ['mse'])


#fitting the ann to the training set
ann = model.fit(X_train, y_train, validation_split = 0.3, batch_size = 30, epochs = 5)
```

```
Epoch 1/5
33/33 [==============================] - 2s 11ms/step - loss: 1104.3722 - mse: 1104.3722 - val_loss: 1673.5154 - val_mse: 1673.5154
Epoch 2/5
33/33 [==============================] - 0s 5ms/step - loss: 1104.3722 - mse: 1104.3722 - val_loss: 1673.5154 - val_mse: 1673.5154
Epoch 3/5
33/33 [==============================] - 0s 6ms/step - loss: 1104.3722 - mse: 1104.3722 - val_loss: 1673.5154 - val_mse: 1673.5154
Epoch 4/5
33/33 [==============================] - 0s 4ms/step - loss: 1104.3722 - mse: 1104.3722 - val_loss: 1673.5154 - val_mse: 1673.5154
Epoch 5/5
33/33 [==============================] - 0s 5ms/step - loss: 1104.3722 - mse: 1104.3722 - val_loss: 1673.5154 - val_mse: 1673.5154
```