**XGBoost classifier**

XGBoost is a machine learning algorithm that belongs to the ensemble learning category, specifically the gradient boosting framework. It utilizes decision trees as base learners and employs regularization techniques to enhance model generalization. Known for its computational efficiency, feature importance analysis, and handling of missing values, XGBoost is widely used for tasks such as regression, classification, and ranking.

**#XGBoost classifier**

import pandas as pd

import numpy as np

df = pd.read\_csv('/content/sample\_data/Pokemon.csv')

df.head()

df.drop(columns=['#','Name',  'Type 1', 'Type 2'],inplace=True)

df

df.Legendary.value\_counts()

# Import label encoder

from sklearn import preprocessing

# label\_encoder object knows how to understand word labels.

label\_encoder = preprocessing.LabelEncoder()

# Encode labels in column 'species'.

df['Legendary']= label\_encoder.fit\_transform(df['Legendary'])

df['Legendary'].unique()

df

df.Legendary.value\_counts()

from sklearn.model\_selection import train\_test\_split

pokemon\_features = df.drop("Legendary",axis=1)

target = df["Legendary"]

X\_train,X\_test,Y\_train,Y\_test = train\_test\_split(pokemon\_features,target,test\_size=0.20,random\_state=0)

from sklearn.metrics import accuracy\_score

import xgboost as xgb

xgb\_model = xgb.XGBClassifier(objective="binary:logistic", random\_state=42)

xgb\_model.fit(X\_train, Y\_train)

Y\_pred\_xgb = xgb\_model.predict(X\_test)

score\_xgb = round(accuracy\_score(Y\_pred\_xgb,Y\_test)\*100,2)

print("The accuracy score achieved using XGBoost is: "+str(score\_xgb)+" %")

from keras.models import Sequential

from keras.layers import Dense

import tensorflow as tf

model = Sequential()

model.add(Dense(32,activation='relu',input\_dim=8))

model.add(Dense(1,activation='sigmoid'))

model.compile(loss='binary\_crossentropy',optimizer='adam',metrics=['accuracy'])

model.fit(X\_train,Y\_train,epochs=100, callbacks = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=3))

Y\_pred\_nn = model.predict(X\_test)

rounded = [round(x[0]) for x in Y\_pred\_nn]

Y\_pred\_nn = rounded

score\_nn = round(accuracy\_score(Y\_pred\_nn,Y\_test)\*100,2)

print("The accuracy score achieved using Neural Network is: "+str(score\_nn)+" %")

