**Star Classifier(raw version)**

**What is it?**

* It is an **UNIQUE** application of **Machine Learning(ML)** in **ASTRONOMY** which can be used to classify stars **(whether it is a Brown Dwarf, Red dwarf, White dwarf, Supergiant or Hypergiant).**

**What it does?**

* It uses an algorithm (Deep Neural network) which learns data of stars from a **CSV file (an excel file format)** that contains data of 200 stars along with its corresponding class.
* **After Learning Data** , this algorithm is used to **test new data** and **predict star types.**

**Background Check of Star classes!**

* Stars in the universe can be classified into several groups . The groups I have used here are :
* **Brown Dwarf**
* **Red Dwarf**
* **White Dwarf**
* **Supergiant**
* **Hypergiant**
* These groups are classified based on several characteristics of stars such as :
* **Luminosity(L/L0) \*\***
* **Radius (R/R0) \*\***
* **Surface Temperature(T/T0) \*\***
* **Absolute Magnitude(Mv)**
* **Spectral Class ( O, B,A,F,G,K,M)**

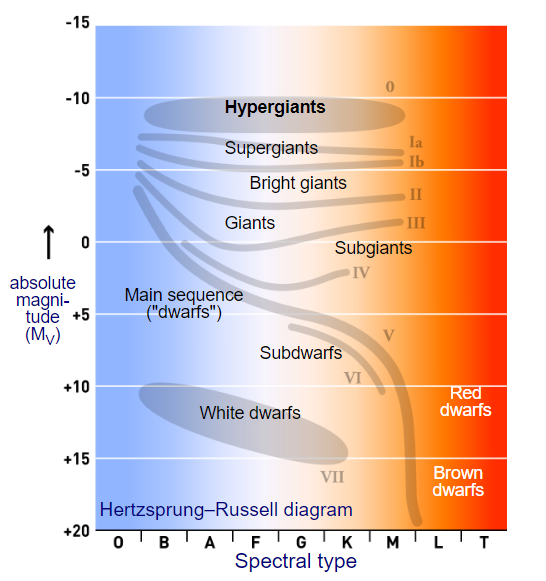
**\*\*** The **Luminosity** , **Radius** and **Surface Temperatures** are taken with respect to that of **Sun’s** . Subscript 0 denotes these values of Sun.

**L0 = 3.828 x 1026 Watts**

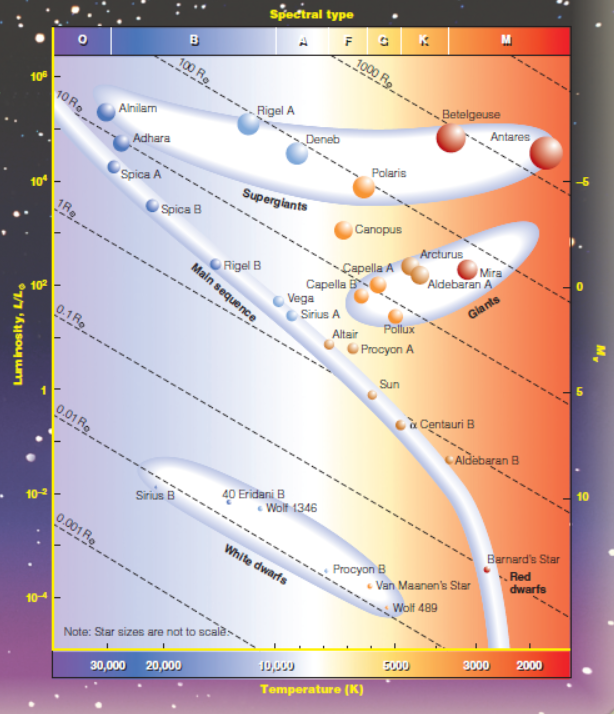
**R0 = 6.9551 x 108 m**

**T0 = 5778 K**

* The classification of stars is done using a H-R Diagram as shown below:



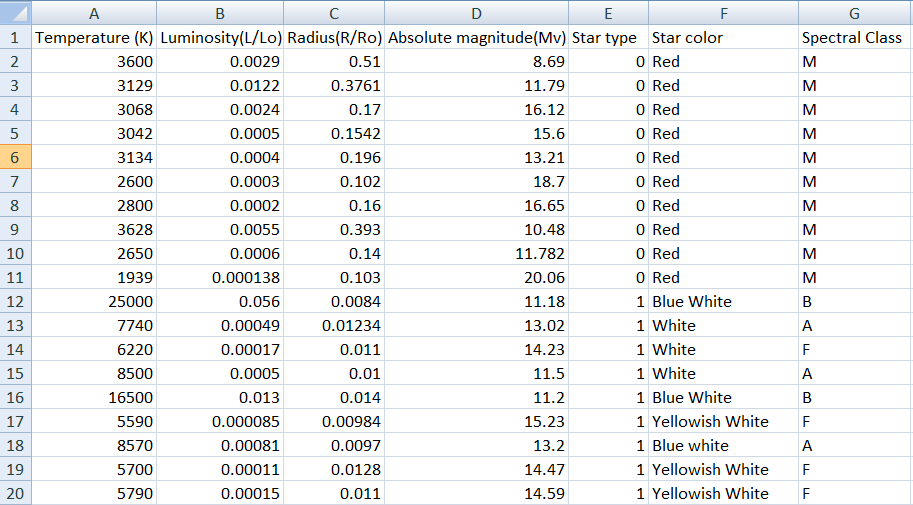
**Fig 1: H-R Diagram 1**

****

**Fig 2: H-R Diagram 2**

**How it works?**

* The dataset of stars are loaded into the program a sample of which is given below:



**Fig 3: A sample Star dataset (CSV file)**

N.B. :This dataset is made by me by collecting data from several sources in the internet and by manual calculations using **Stefan-Boltzmann’s law of black body radiations**(**finding luminosity**) and **Stefan’s law (finding surface temperature).**

* In the dataset, “**Star Type**” denotes **:**

**0 – Brown Dwarf**

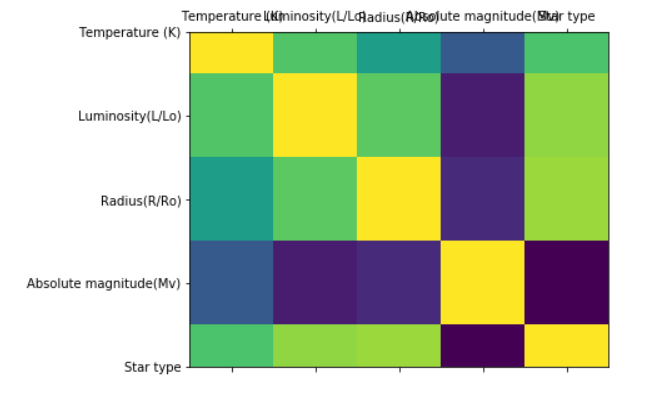
**1 – Red Dwarf**

**2 – White Dwarf**

**3 – Supergiant**

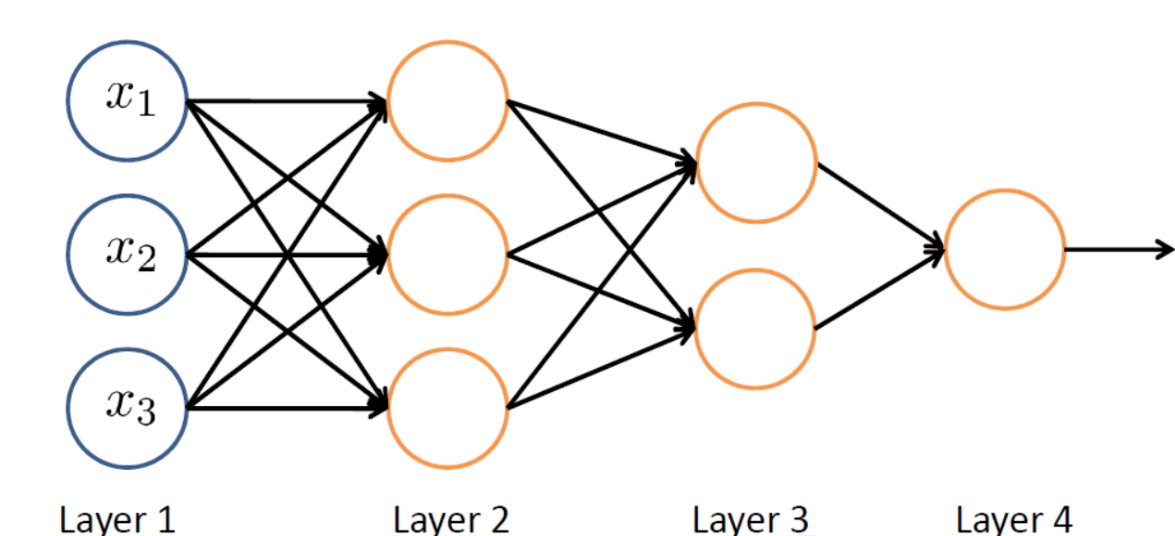
**4 – Hypergiant**

* We check for **redundant column(removed if found)** by analysing the **correlation** and plotting it in a graph which is shown below:



N.B. :- Existence of yellow square boxes outside the diagonal indicates redundant column(in our case , no such column found)

* The **first 4 columns of the dataset** is fed into an algorithm called **Deep Neural Network** which looks like the fig. below:

****

**Input data Hidden layers Output data**

**Fig 4: A Deep Neural Network**

* **Input data:** It is the data of the **first 4 columns** of the dataset which it uses as input .
* **Output data:** It consists of the **“Star Type”** column in the dataset.
* **Hidden Layers :** These layers are used to find a **mapping function** that maps **input data with output data**. This process is called **Training** or **Learning.**

**# A simple example to get the feel of it:**

Say, A = 1,2,3,4,5,6,7,8,9,10

B = 10,20,30,40,50,60,70,80,90,100.

We can see that the mapping function is

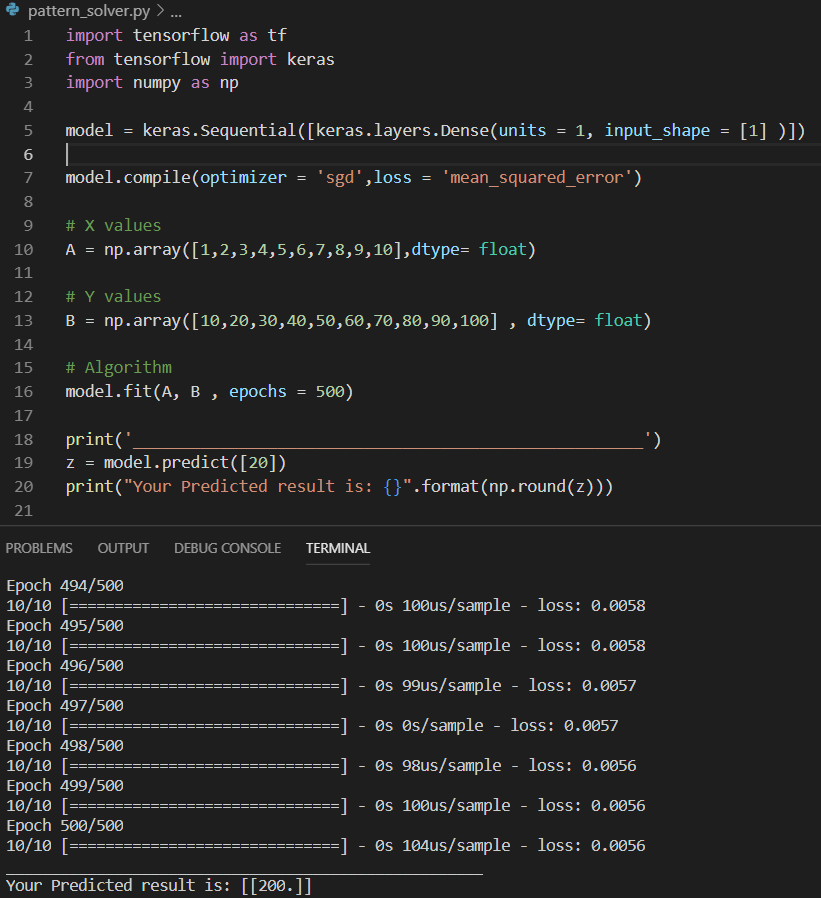
**B = 10 x A**

Now, what if your PC learns this data and predicts : if A = 20, then B = ?

Definitely, B = 10 x 20 = 200

**What if your PC finds the mapping function?**

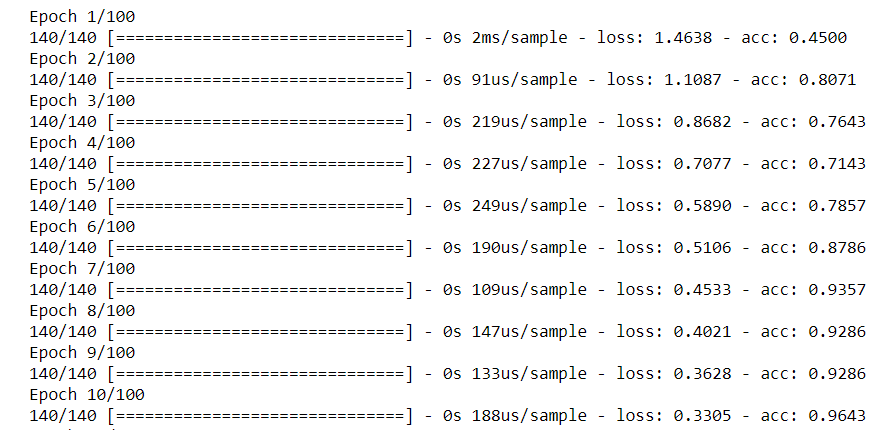
Kidding right??...Well…See below how it does it

****

**Fig 5: A simple application of ML**

# **Switching Back to our previous discussion**

* The Algorithm I used learns 70% of the star dataset (randomly distributed data) . The learning takes place as follows:



**Fig 6: Process of Training Data**

* Here “Epoch” denotes the number of iterations completed.
* “loss” and “accuracy” are the factors which indicates how well the algorithm learned the data.

**How it Predicts Star Classes?**

* From the 70% data it learned, remaining 30% data are tested and the algorithm predicts star types in the form of an array:

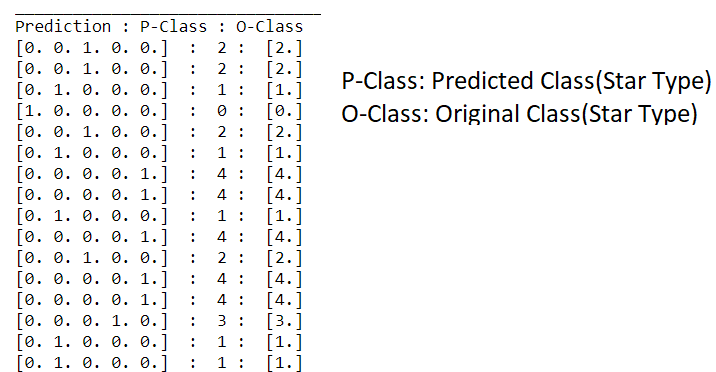
[1,0,0,0,0] denotes **class 0** : White dwarf

[0,1,0,0,0] denotes **class 1** : Red dwarf

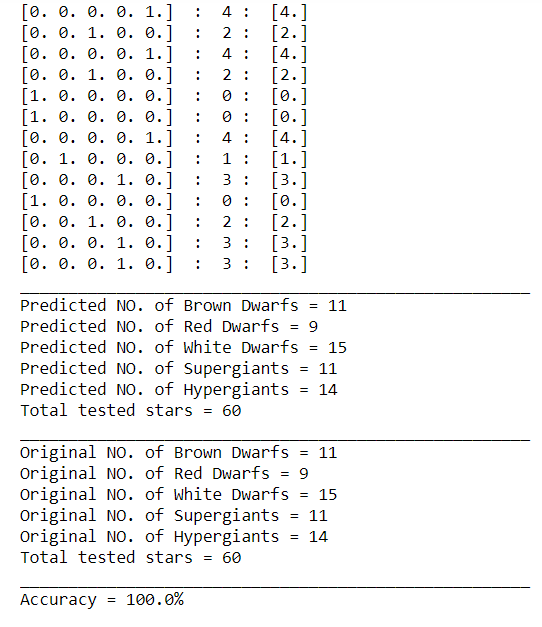
[0,0,1,0,0] denotes **class 2**: White dwarf

[0,0,0,1,0] denotes **class 3**: Supergiant

[0,0,0,0,1] denotes **class 4**: Hypergiant



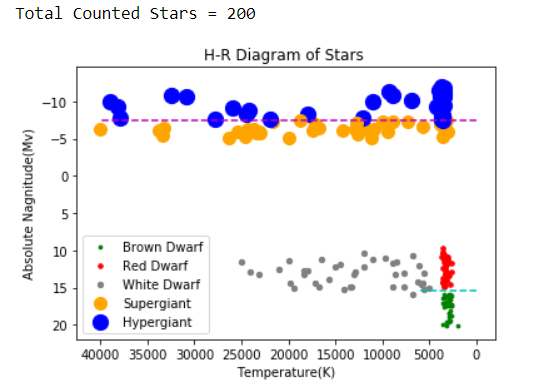
**Fig 7: Predicting Star types of tested data**

****

**Fig 8: Predicting Star Types with accuracy**

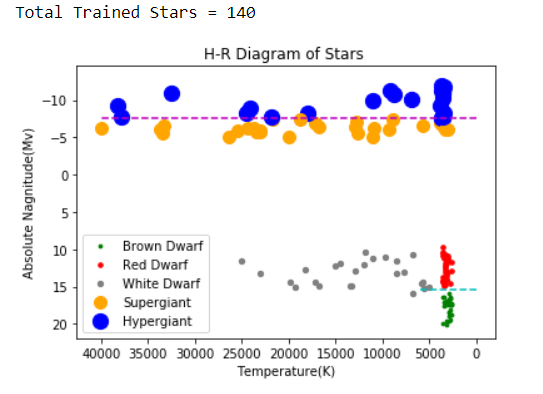
**How to Visualize the data?**

* Data visualization is a key aspect of ML .
* First we need to know that most of the stars in the universe can be classified based on the H-R Diagram of stars shown previously.
* Let’s check out the H-R Diagram for all the stars in our dataset.

****

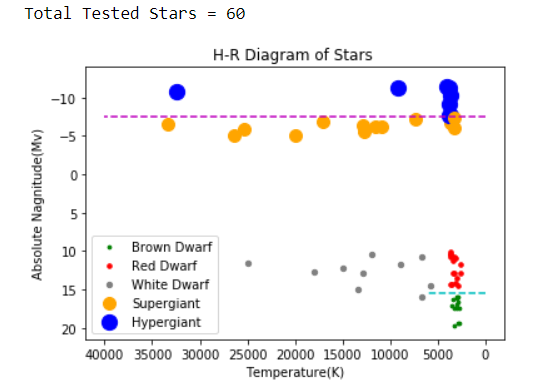
**Fig 9: H-R diagram of whole Dataset**

* From the 200 stars, we trained 70 % (i.e. 140) stars randomly . The H-R Diagram for trained stars looks like:

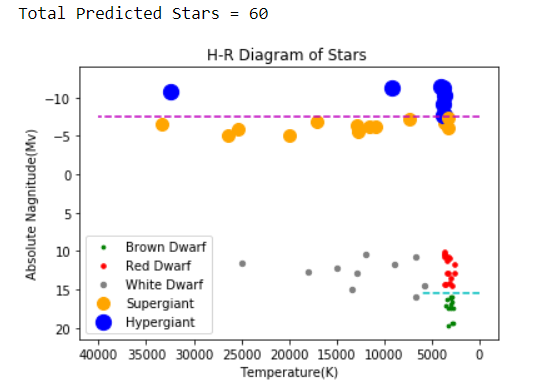
****

**Fig 10: H-R Diagram of Trained stars**

* After analysing the original tested data, if the predicted data is almost similar (at least 98% accurate) to that of the tested dataset then we can say that the classifier learned efficiently (In our case we got 100% accuracy).



**Fig 11: H-R Diagram of Original tested stars**

****

**Fig 12: H-R Diagram of Predicted stars**

* Clearly, we can see that both the above H-R diagrams are equivalent and hence it learned well

consequently predicted with utmost accuracy(100% in our case).

**What is its utility?**

Viewers reading this part already get the utility of the model as this when **fed with future data** can **predict the type of star** with utmost accuracy and the **best part is that it will take just a few seconds** to predict the **types of thousands of stars** **all at once** without **manually predicting** one by one by scientists.

**N.B. :- More the data u train, better it will predict**

**\*\*\*\*\*\*\*\*\*\*\* Thanks For Reading \*\*\*\*\*\*\*\*\*\*\*\*\***

**Sources:**

* **Wikipedia**
* **Stars and Galaxies by SEEDS | Backman**