Neural_FCA

Ruben Safaryan

December 13, 2022

About the Dataset

Star Type Classification

Temperature – K
L – L/Lo (Luminosity)
R – R/Ro (Raduis)
AM – Mv (Absolute Magnitude)
Color – General Color of Spectrum
Spectral_Class – O,B,A,F,G,K,M
TARGET: Type
Red Dwarf - 0
Brown Dwarf - 1
White Dwarf - 2
Main Sequence - 3
Super Giants - 4
Hyper Giants - 5

The dataset contains data on 240 stars. It is available openly on Kaggle.

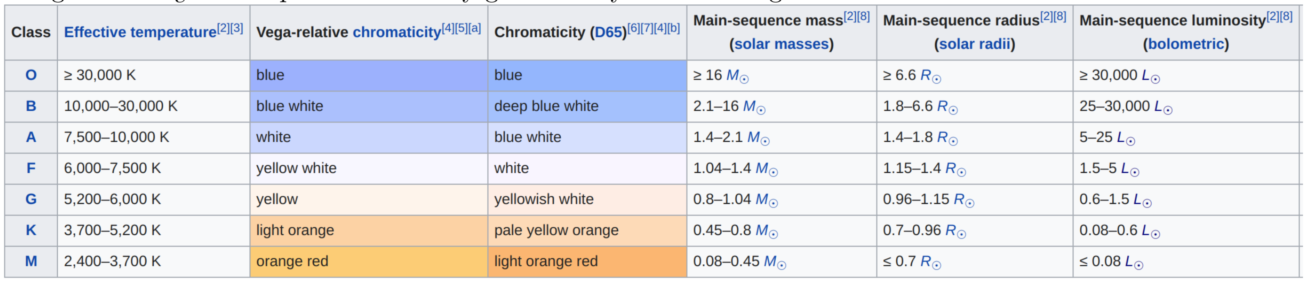
The temperature is given in Kelvin. The luminosity and radius are given relative to that of the Sun.

It is well known that most stars are small even in comparison to the Sun, however in the Star Type Classification dataset a good balance is kept in terms of variety of entries. In addition, to remain aligned with the task at hand, the Type attribute will be reduced to a binary variable, indicating weather a star is a dwarf or not.

Preprocessing

The categorical variables such as Color and Spectral_Class will be dealt with using OneHotEncoding, while setting a minimal frequency to absorb some entries into the insignificant class.

For the continuous variables I chose to set up bins according to the average properties of different types of stars. First a binary label is assigned for each entry falling into a certain bin. Later the bins are merged with "¿". This process is mostly governed by the following table:

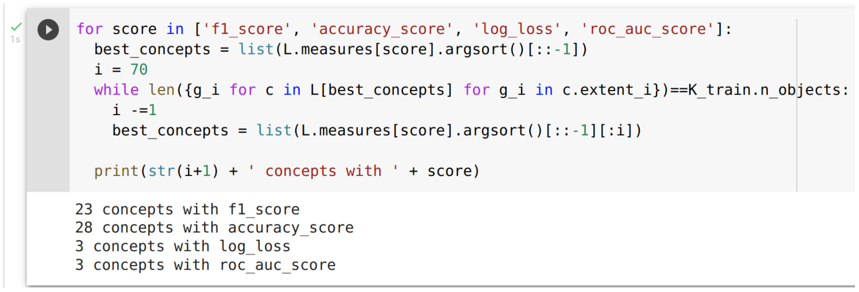


Class	Effective temperature[2][3]	Vega-relative chromaticity[4][5][a]	Chromaticity (D65)[6][7][4][b]	Main-sequence mass[2][8] (solar masses)	Main-sequence radius[2][8] (solar radii)	Main-sequence luminosity[2][8] (bolometric)
O	≥ 30,000 K	blue	blue	≥ 16 $M_\odot$	≥ 6.6 $R_\odot$	≥ 30,000 $L_\odot$
B	10,000–30,000 K	blue white	deep blue white	2.1–16 $M_\odot$	1.8–6.6 $R_\odot$	25–30,000 $L_\odot$
A	7,500–10,000 K	white	blue white	1.4–2.1 $M_\odot$	1.4–1.8 $R_\odot$	5–25 $L_\odot$
F	6,000–7,500 K	yellow white	white	1.04–1.4 $M_\odot$	1.15–1.4 $R_\odot$	1.5–5 $L_\odot$
G	5,200–6,000 K	yellow	yellowish white	0.8–1.04 $M_\odot$	0.96–1.15 $R_\odot$	0.6–1.5 $L_\odot$
K	3,700–5,200 K	light orange	pale yellow orange	0.45–0.8 $M_\odot$	0.7–0.96 $R_\odot$	0.08–0.6 $L_\odot$
M	2,400–3,700 K	orange red	light orange red	0.08–0.45 $M_\odot$	≤ 0.7 $R_\odot$	≤ 0.08 $L_\odot$

Quality Measure

The data in the dataset does not restrict the use of most metrics, as in this case the importance of a True Positive is not greater than of a True Negative.(As opposed to, for example, classifying an illness) So we are free to experiment and find a measure that is simultaneously suitable and beneficial for the method proposed in the task.

Trying some well-known scores we can compare the minimal number of concepts that cover all objects in the train data.



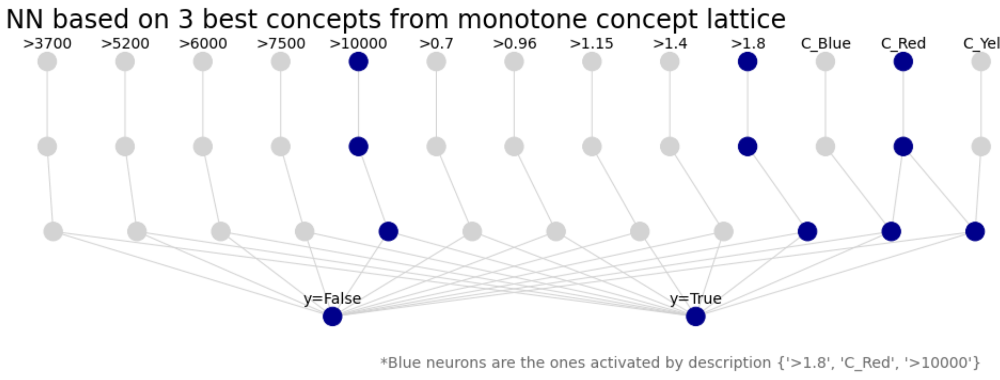
```
for score in ['f1_score', 'accuracy_score', 'log_loss', 'roc_auc_score']:
  best_concepts = list(L.measures[score].argsort()[::-1])
  i = 70
  while len({g_i for c in L[best_concepts] for g_i in c.extent_i})==K_train.n_objects:
    i -=1
    best_concepts = list(L.measures[score].argsort()[::-1][:i])

  print(str(i+1) + ' concepts with ' + score)
```

```
23 concepts with f1_score
28 concepts with accuracy_score
3 concepts with log_loss
3 concepts with roc_auc_score
```

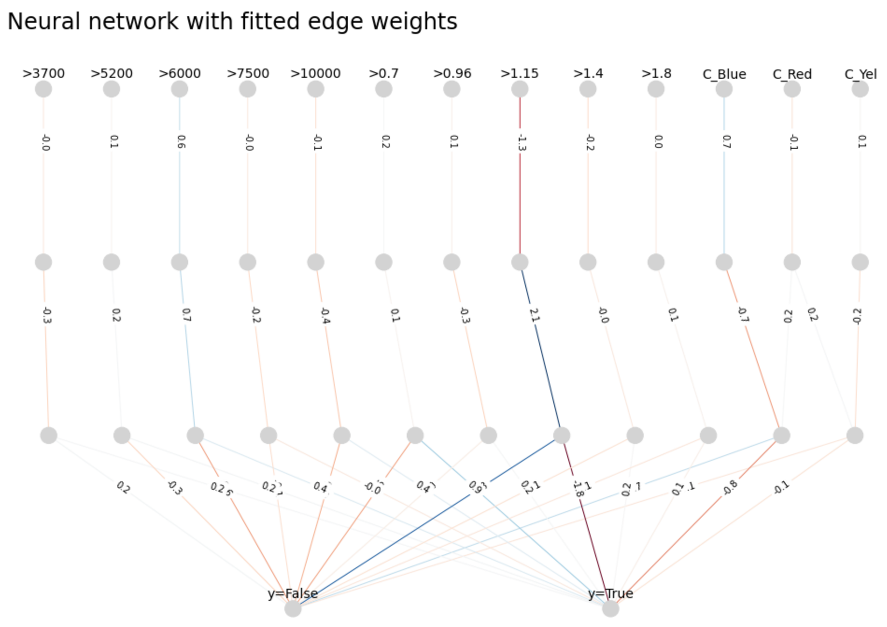
Clearly log_loss and auc_roc are to be favored here as ways of choosing best concepts. For performance evaluation we will stick with accuracy and f1.

Fit and Visualize the Network

First we find best concepts by assuming that an object is predicted True if it is in the extent of a concept, and calculating the prediction score according to some metric. The metric of choice is log_loss, which gives better results as shown in the code.

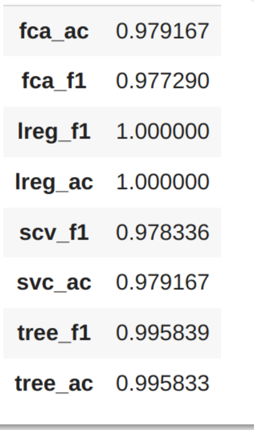

NN based on 3 best concepts from monotone concept lattice
>3700
>5200
>6000
>7500
>10000
>0.7
>0.96
>1.15
>1.4
>1.8
C_Blue
C_Red
C_Yel
y=False
y=True
*Blue neurons are the ones activated by description {'>1.8', 'C_Red', '>10000'}

After fitting we can visualize the resulting network with the weights and vertices



Comparison to other methods

Here we try fitting and evaluating other classifiers to compare with the approach given by the task.
Here are the results:



fca_ac	0.979167
fca_f1	0.977290
lreg_f1	1.000000
lreg_ac	1.000000
scv_f1	0.978336
svc_ac	0.979167
tree_f1	0.995839
tree_ac	0.995833

The methods used are Logistic Regression, SCV and Decision Trees.
It is clear that because the dataset is small most models overfit.