CYO - Star Classification

Andres Antury

11/02/2022

1. Executive summary

The aim of this project is to create a start classification system. The dataset used for this project is a 6-class star dataset for star classification which was sourced from *(https://www.kaggle.com/deepu1109/star-dataset/metadata)*. The dataset was licensed by its author Deepraj Baidya. The dataset included the following variables:

- Absolute Temperature (in K) consisting on the surface temperature of different stars
- Relative Luminosity (L/Lo) of the starts compared with the Sun
- Radius (R/Ro) of the starts compared with the Sun
- Absolute Visual Magnitude of Stars
- Star Type which is divided in 6 classes *(Brown Dwarf, Red Dwarf, White Dwarf, Main Sequence star, Supergiant and Hypergiant)*
- Star Color obtained following spectral analysis
- Spectral Class which is divided in 6 classes *(O,B,A,F,G,K,M)*

The process included data cleansing, data exploration as well as feature selection to ensure only relevant data was used for the different algorithms used.

In order to create the Machine Learning algorithm for the classification system, the star dataset was split using the Caret package to create the the training and test sets. The evaluation metrics used to achieve the best model were derived using the accuracy which is developed using the Confusion Matrix. *(Irizarri, R. Introduction to Data Science, p. 506).*

The R packages used for this project included *"tidyverse"*, *"caret"*, *"data.table"*, *"randomForest"*, *"nnet"*, *"party"*, and *"earth"*.

The models compared were trained using the caret package and included *Random Forest*, *Conditional Inference Random Forest*, *Bagged Multivariate Adaptive Regression Splines*, and *Penalized Multinomial Regression.*

A comparison table was created to view the different accuracy values achieved by the models trained with the train set and then tested with the test set.

2. Analysis

For the analysis, the data set is downloaded from the original repository and the csv file saved in the personal Github repository of the author of this report which can be found on *(https://raw.githubusercontent.com/aantury/CYO_Star-Classification/main/6_class.csv*. The Dataset is retrieved from the Github repository using the methodology introduced during the course *(Irizarri, R. Introduction to Data Science, p. 109)*

The 6_class.csv dataset once downloaded into the R script is then converted to a data.frame and renamed as "start_dat".

2.1 Data exploration and data cleansing

Data exploration and data cleansing were conducted in order for the data to be formatted in a way that allowed better interpretation and usability.

To view the top 6 rows of star_dat dataset and the column names we use the "head" function on the dataset.

```
head(star_dat)
```

```
# A tibble: 6 x 7
  Temperature Luminosity Radius Absolute_magnitude Star_color Spectral_class
        <dbl>      <dbl>  <dbl>              <dbl> <chr>      <chr>
1        3068   0.0024    0.17                16.1 Red        M
2        3042   0.0005    0.154               16.6 Red        M
3        2600   0.0003    0.102               18.7 Red        M
4        2800   0.0002    0.16                16.6 Red        M
5        1939   0.000138  0.103               20.1 Red        M
6        2840   0.00065   0.11                17.0 Red        M
# ... with 1 more variable: Star_type <dbl>
```

Using the "nrow" function we can confirm that the number of observations (rows) in the star_dat dataset is:

```
[1] 240
```

Star_type has nominal numbers assigned instead of descriptive names. These will be replaced by the actual star type for better interpretation e.g Star type 0 = Brown Dwarf, etc., as specified in the original dataset repository using the following code (only 1 star type shown):

```
star_dat$Star_type <- as.character(star_dat$Star_type)
star_dat["Star_type"][star_dat["Star_type"] == 0] <- "Brown_Dwarf"
```

The dataset structure can be viewed using the "str" function and it can be seen that Temperature, Luminosity, Radius and Absolute magnitude are numeric while Star color, Spectral Class and Star type are character as expected.

To find the number of occurrences of different star colors we use the "table" function. This gives us the ranking of the colors according to the number of times they appear in the dataset

```
                Var1 Freq
1                Red  112
2               Blue   56
3         Blue-white   26
```

```
4          Blue White   10
5        yellow-white    8
6               White    7
7          Blue white    4
8               white    3
9     Yellowish White    3
10             Orange    2
11            Whitish    2
12          yellowish    2
13         Blue-White    1
14         Orange-Red    1
15 Pale yellow orange    1
16       White-Yellow    1
17          Yellowish    1
```

We can see that the Star_color column has duplicate entries for the same color (with slight differences in spelling), those repeated colors are be merged and presented with the same formatting using the following code (only 1 color type shown):

```
star_dat[star_dat == "Blue-white" | star_dat == "Blue White"| star_dat == "Blue white" |
           star_dat == "Blue-White" ] <- "Blue_White"
```

Similarly, to obtain number of occurrences of the different spectral classes we use the "table" function. We can see that the Spectral class most frequent in the dataset is the spectral class M

```
  Var1 Freq
1    M  111
2    B   46
3    O   40
4    A   19
5    F   17
6    K    6
7    G    1
```

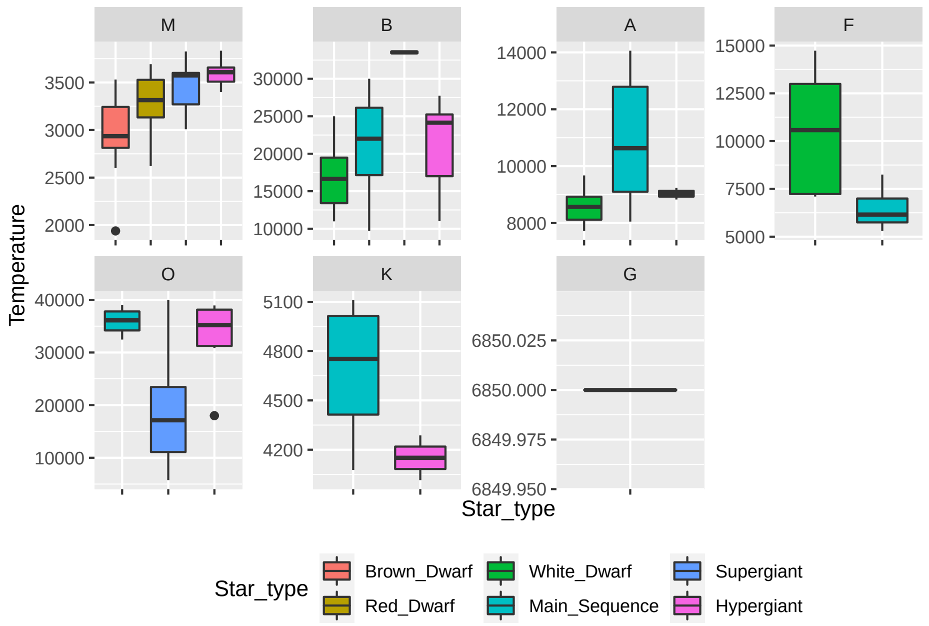
Converting the character columns to factors. This is important for the analysis of the models to be used later on. The methodology was adapted from *(https://michaelbarrowman.co.uk/post/convert-all-character-variables-to-factors/)*

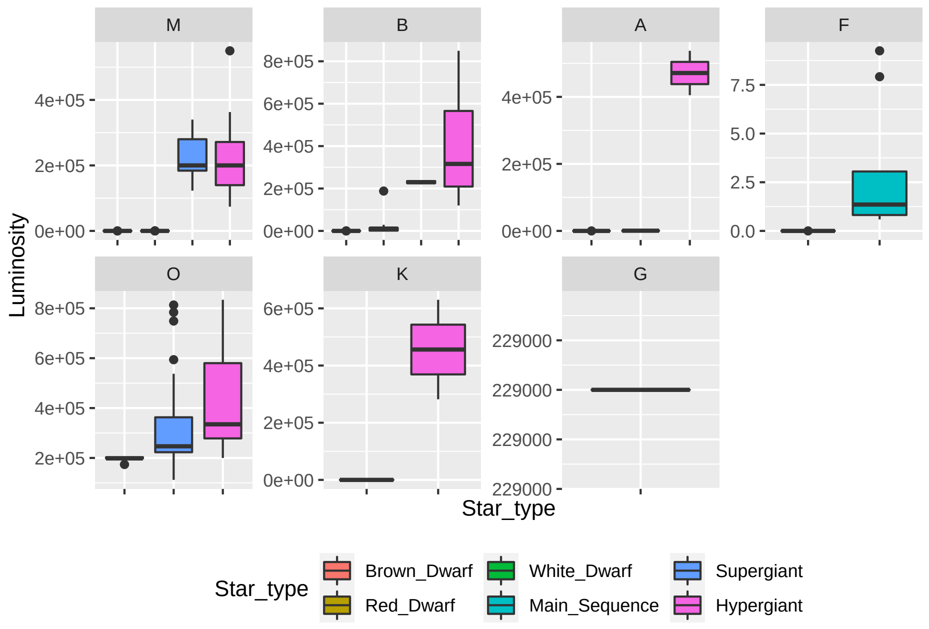
```
star_dat <- star_dat %>% mutate(across(where(is.character),as_factor))

head(star_dat)
```

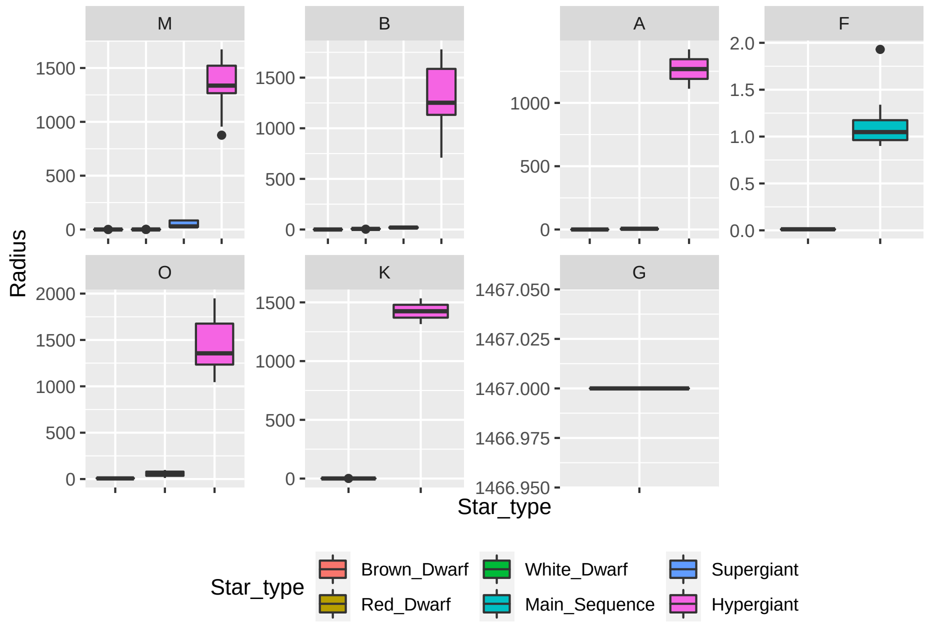
```
# A tibble: 6 x 7
  Temperature Luminosity Radius Absolute_magnitude Star_color Spectral_class
        <dbl>      <dbl>  <dbl>              <dbl> <fct>      <fct>
1        3068   0.0024    0.17                16.1 Red        M
2        3042   0.0005    0.154               16.6 Red        M
3        2600   0.0003    0.102               18.7 Red        M
4        2800   0.0002    0.16                16.6 Red        M
5        1939   0.000138  0.103               20.1 Red        M
6        2840   0.00065   0.11                17.0 Red        M
# ... with 1 more variable: Star_type <fct>
```

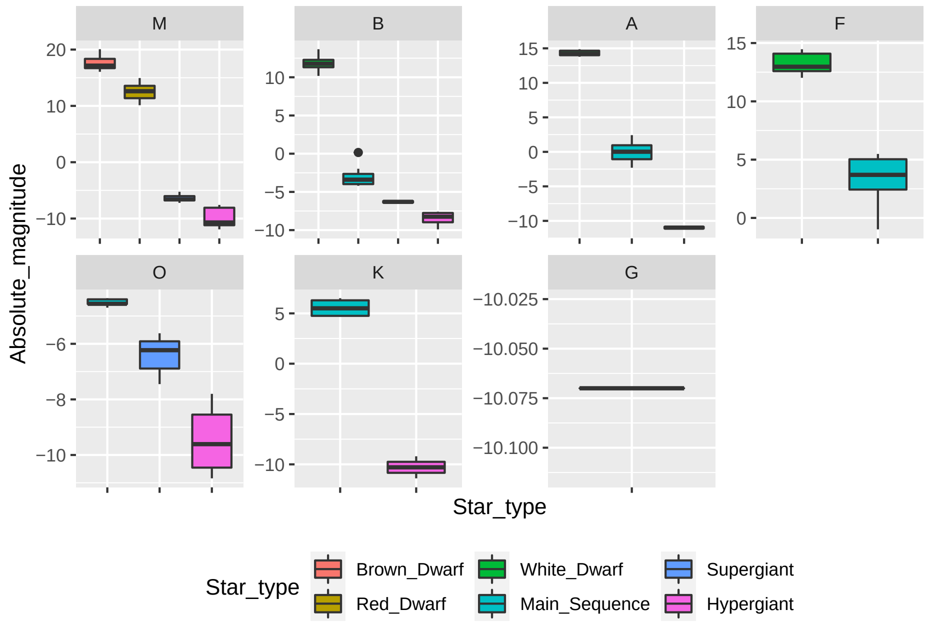
2.2 Data Visualization

To view the different features plotted against Star type and wrapped around Spectral Class (M,B,A,F,O,K,G), we can use the boxplot diagram as below:









We can see that spectral class G gives very little information and this could be due to the very small number sampled for this class. it can also be seen that the radius of the Hypergiants is considerably larger than the other types as expected. The boxplot allows to identify the most relevant features per star type and per spectral class. Further information regarding the relationship between the different features can be seen later in the correlation analysis.

2.3 Feature Selection

2.3.1 Correlation

a. Converting the star_dat dataset into a matrix. Only the continuous features are selected.

```r
star_dat_cM <- star_dat %>% select(Temperature, Luminosity, Radius,
                                   Absolute_magnitude)%>% as.matrix()
```

b. Creation of correlation matrix using the "cor" function. Only showing 2 decimals for ease of viewing.

```r
CM <- round(cor(star_dat_cM), 2)
```

```
                   Temperature Luminosity Radius Absolute_magnitude
Temperature               1.00       0.39   0.06              -0.42
Luminosity                0.39       1.00   0.53              -0.69
Radius                    0.06       0.53   1.00              -0.61
Absolute_magnitude       -0.42      -0.69  -0.61               1.00
```

c. To see whether objects are highly correlated and what are they, we use the "findCorrelation" function.

```r
high_corr <- findCorrelation(CM, cutoff=0.5, exact = TRUE)
```

```
[1] 4 2
```

As can be seen from the table, both Luminosity (2) and Absolute Magnitude (4) are highly correlated with each other, therefore, these features should not be used for analysis.

The features that will be used as predictors are: **Temperature**, **Radius**, **Star Color** and **Spectral Class**. The target feature is **Star type**

2.3.2 Near-zero Variance In addition to the correlation, it is useful to check whether any of the features in de dataset have near Zero variance. It is important to find and remove any zero or near-zero variance features as they add no value to the analysis. In this case the "nearZeroVar" function from the caret package is used. Methodology used was obtained from *(https://www.rdocumentation.org/packages/caret/versions/6.0-90/topics/nearZeroVar)*

```r
star_dat_nzv <- nearZeroVar(star_dat, saveMetrics = TRUE)
```

```
                   freqRatio percentUnique zeroVar   nzv
Temperature         1.500000     95.000000   FALSE FALSE
Luminosity          1.666667     86.666667   FALSE FALSE
Radius              1.000000     90.000000   FALSE FALSE
Absolute_magnitude  1.500000     95.000000   FALSE FALSE
Star_color          2.000000      5.000000   FALSE FALSE
Spectral_class      2.413043      2.916667   FALSE FALSE
Star_type           1.000000      2.500000   FALSE FALSE
```

Using the "is.na" function in the star_dat data set, it can be confirmed that there are none of the features within the dataset have any NA values.

2.4 Train and Test sets

The methodology introduced during the course will be used to obtain the test and train datasets for this CYO project. The "createDataPartition" function of the "caret" package is used to create the train and test sets.

The Test set will be 20% of star_dat dataset. This percentage of partition was chosen because the star_dat dataset is not very large as there are 240 observations.

```
set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(y = star_dat$Star_type, times = 1, p = 0.2, list = FALSE)
train_data <- star_dat[-test_index,]
test_data <- star_dat[test_index,]
```

2.5 Machine Learning Algorithms Training and Testing

For this project, the comparison between different machine learning models will be using the "train" function of the caret package. The target feature is the **Star_type**, therefore, the Star Type on the test set needs to be converted to factor in order to be used to test the models.

```
test_data$Star_type <- as.factor(test_data$Star_type)
```

As explained earlier, the predictors to be used are Temperature, Radius, Star Color and Spectral Class.

The modeling techniques used for this project were: **Random Forest**, **Conditional Inference Random Forest**, **Bagged Multivariate Adaptive Regression Splines (MARS)**, and **Penalized Multinomial Regression**.

The methodology used for the different models was based on the caret documentation found on *(https://topepo.github.io/caret/train-models-by-tag)*.

2.5.1 Random Forest using caret method "rf" and the "randomForest" package as explained in *(https://topepo.github.io/caret/train-models-by-tag.html#Random_Forest)*

a. Training the model using default "trainControl" option

```
star_rf <- train(Star_type ~ Temperature + Radius +
                           Star_color + Spectral_class, method = "rf", data=train_data,
                         metric = "Accuracy", trControl = trainControl())
```

b. Model prediction using the "predict" function on the test_data set

```
star_prediction_rf <- predict(star_rf, test_data)
```

c. Accuracy as per the Confusion Matrix

```
confusionMatrix(star_prediction_rf, test_data$Star_type)$overall[["Accuracy"]]
```

2.5.2 Conditional Inference Random Forest. Implementation of the random forest and bagging ensemble using the using caret method "cforest" and the "party" package found in *(https://www.rdocumentation.org/packages/partykit/versions/1.2-15/topics/cforest)* as explained in *(https://topepo.github.io/caret/train-models-by-tag.html)*

a. Training the model using the default train control values in "cforest_control"

```
star_cforest <- train(Star_type ~ Temperature + Radius +
                      Star_color + Spectral_class, method = "cforest",
                      data=train_data, metric = "Accuracy", controls = cforest_control() )
```

b. Model prediction using the "predict" function on the test_data set

```
star_prediction_cforest <- predict(star_cforest, test_data)
```

c. Accuracy as per the Confusion Matrix

```
confusionMatrix(star_prediction_cforest, test_data$Star_type)$overall[["Accuracy"]]
```

2.5.3 Bagged Multivariate Adaptive Regression Splines (MARS) using using caret method "bagEarth" and the "earth" package found in *(https://www.rdocumentation.org/packages/caret/versions/6.0-90/topics/bagEartha)* as explained in *(https://topepo.github.io/caret/train-models-by-tag.html)*

a. Training the model setting "glm option = null" as this is not applicable on this model and would otherwise affect the result

```
star_bagEarth <- train(Star_type ~ Temperature + Radius +
                       Star_color + Spectral_class, method = "bagEarth",
                       data=train_data, metric = "Accuracy", glm = NULL)
```

b. Model prediction using the "predict" function on the test_data set

```
star_prediction_bagEarth <- predict(star_bagEarth, test_data)
```

c. Accuracy as per the Confusion Matrix

```
confusionMatrix(star_prediction_bagEarth, test_data$Star_type)$overall[["Accuracy"]]
```

2.5.4 Neural network with principal component step using caret method "multinom" and the "nnet" package and the penalized multinomial regression found in *(https://www.rdocumentation.org/packages/nnet/versions/7.3-17/topics/multinom)*as explained in *(https://topepo.github.io/caret/train-models-by-tag.html)*

a. Training the model setting the "maxit" parameter to 1000 iterations in order for the model to converge as the default 100 iterations are not sufficient

```r
star_pmr <- train(Star_type ~ Temperature + Radius +
                  Star_color + Spectral_class, method = "multinom", data=train_data,
                  maxit = 1000)
```

b. Model prediction using the “predict” function on the test_data set

```r
star_prediction_pmr <- predict(star_pmr, test_data)
```

c. Accuracy as per the Confusion Matrix

```r
confusionMatrix(star_prediction_pmr, test_data$Star_type)$overall[["Accuracy"]]
```

3. Results

The following table shows the accuracy achieved by the models trained and tested and shows the accuracy achieved by the best performing model which was the one using the **Penalized Multinomial Regression with neural network**, the accuracy achieved was **0.9583**

Method	Accuracy
a. Random Forest	0.9375000
b. Conditional Inference Random Forest	0.9166667
c. Bagged MARS Model	0.8333333
d. Penalized multinomial regression	0.9583333

4. Conclusion

4.1 Summary

The project consisted of 3 main parts:

4.1.1 Data Exploration and Dataset Creation This step included the review of the star classification dataset. The Datasets necessary for the data analysis and Machine Learning model creation were obtained from the start_dat dataset. The datasets created included a partition of the start_dat dataset to create the training and test sets.

4.1.2 Data Analysis Following the creation of the training and test datasets, 4 different models were trained and tested using different techniques including the implementation of the Random Forest and bagging Ensemble, Multivariate Adaptive Regression Splines and Penalized Multinomial Regression. The evaluation metrics used for the models was the accuracy which was obtained from the confusiuon matrix. The accuracy was used to compare the models and to select the best performing model.

4.1.3 Best performing model selection The best performing model based on the test set was the one using **Penalized Multinomial Regression** with Neural Networks which was modeled using the "nnet" package and the "multinom" function and trained in the "caret" package. This model then achieved an accuracy of **0.9583**, leading to a satisfactory result.

4.2 Limitations

Due to the size of the original dataset, the models evaluated could be trained using the "train" function of the "caret" package and using in most cases default parameters. The main driver for this was the preference of the author of this report to provide simple and less complex models.

The main objective has been to reach the goal of selecting the best performing model among the different Machine Learning modeling techniques, and the application of different methodologies learned during the whole data science course.

Should larger datasets become necessary, the models could take a considerable amount of time to run, so it is preferable to limit the size of the datasets to a level similar to the one used for this project if using the "train" function of the "caret" package to train the models.

Alternatively, more complex neural networks based models could be used in larger datasets if required.

4.3 Future work

It would be beneficial to improve the models evaluated so more tuning parameters are used specially on the Penalized Multinomial Regression and the Multivariate Adaptive Regression Splines. This could lead to more robust results and to enable the use of even larger and more complex datasets.

Although there is room for improvement regarding the models, the author of this report believed that the main objective of the project has been achieved and the final result met the expectations.