BMAAR Project Report

Star Classification and Prediction

Star Prediction Dataset

Abstract

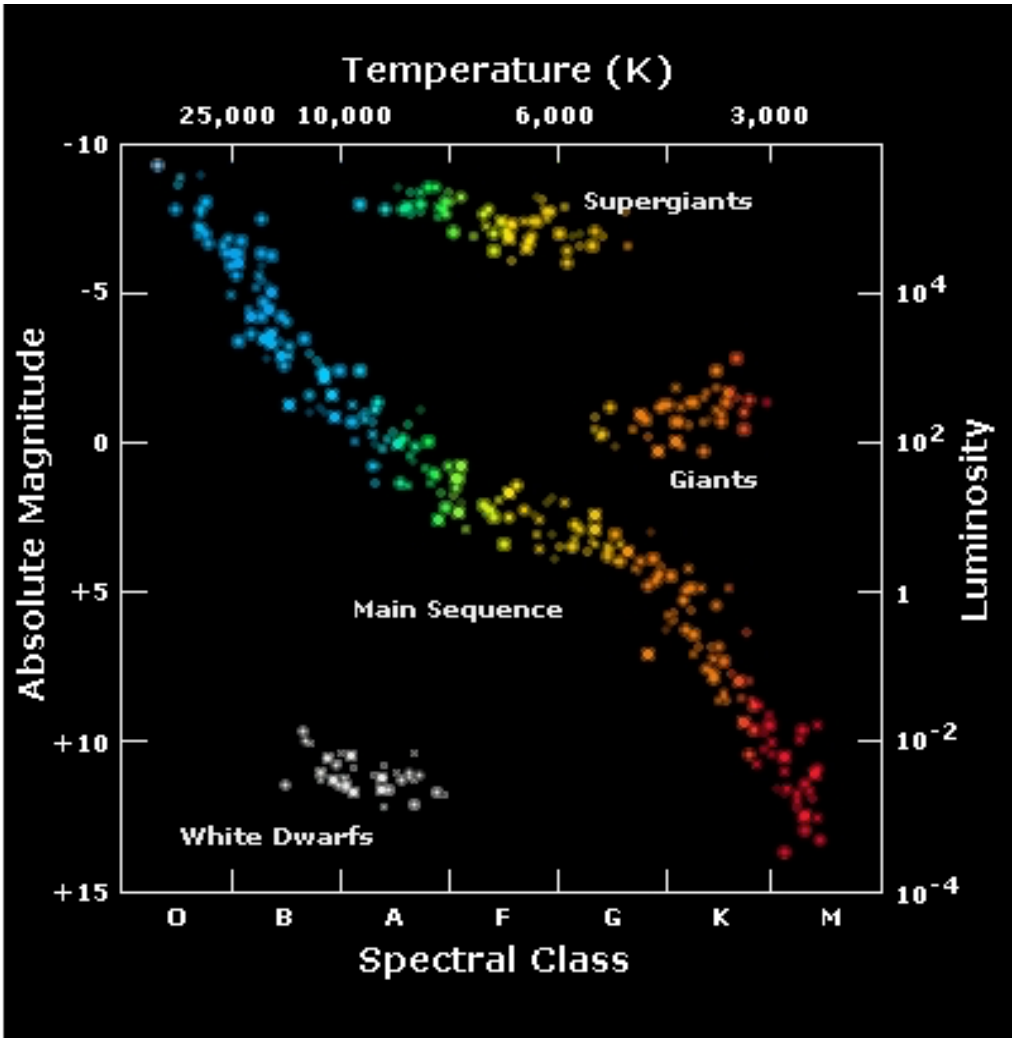
Here we have used a dataset that has features of Hertzsprung–Russell Diagram. Objective of the project is to classify and predict the given stars according to their type (brown dwarf, red dwarf, white dwarf, main sequence, super giants). It has a total of 7 features - Absolute Temperature, Relative Luminosity, Relative Radius, Absolute Magnitude, Star colour, Spectral class, Star type. With these we will initially build different machine learning models and determine which is the most accurate one.

Problem Area

The Hertzsprung–Russell diagram is a scatter plot of stars that depicts the relationship between absolute magnitudes, luminosities, spectral class and temperatures. The purpose of making the dataset is to show that the stars follows a certain pattern in the Space, specifically called HR-Diagram so that we can classify stars by plotting its features based on that graph.

Every star, depending on its initial mass, goes through different evolutionary stages based on its internal structure and how it generates energy. Each of those stages corresponds to a change within the temperature and luminosity of

the star, which may be seen to move to different regions because it evolves. With the help of this astronomers can know the internal structure and evolutionary stages of stars just by determining its position within the diagram.



About the dataset

Instances – 240

Features – 7 (Categorical – 3, Numerical - 4)

No missing data

Features:

1. Temperature (K)

2. Luminosity (L/Lo)

3. Radius (R/Ro)

4. Magnitude (Mv)

5. Star Colour (White, Blue, Yellow, Orange, Red, ….)

6. Spectral Class (A, B, F, G, K, M, O)

7. **Star Type (target class)**

0 -> Brown Dwarf, 1 -> Red Dwarf, 2 -> White Dwarf,

3 -> Main Sequence, 4 -> Super Giants, 5 -> Hyper Giants

Problem Statement

Here we are given with a dataset that has HRD features. Our aim is to classify and predict them effectively in order to predict the type of each star using different machine learning algorithms. Just finding most accurate model won't make the cut sometimes. Because some models works better for some types of data while other don't. Some might show inconsistent results. So we have to take these factors into the consideration too.

Choice of Methodology and Algorithm

Our problem is multi class classification. We are trying to predict type of star which is classified in 6 different categories. Some of the best machine learning algorithms for this type of problems are KNN, Naïve Bayes, Decision Tree, SVM, Random Forest etc.

In this project we use the following machine learning algorithms:

K-Nearest Neighbor (KNN):

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.

K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.

K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.

K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.

K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data.

It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.

KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

Naïve Bayes (NB):

Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.

It is mainly used in *text classification* that includes a high-dimensional training dataset.

Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.

It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.

Some popular examples of Naïve Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles.

Decision Tree:

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving

Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

The decisions or the test are performed on the basis of features of the given dataset.

It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.

It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.

In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm.

Support Vector Machine (SVM):

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

Descriptive Analytics

Dataset:

	Temperature..K.	Luminosity.L.Lo.	Radius.R.Ro.	Absolute.magnitude.Mv.	Star.type	Star.color	Spectral.Class
1	3068	2.400e-03	1.700e-01	16.120	0	Red	M
2	3042	5.000e-04	1.542e-01	16.600	0	Red	M
3	2600	3.000e-04	1.020e-01	18.700	0	Red	M
4	2800	2.000e-04	1.600e-01	16.650	0	Red	M
5	1939	1.380e-04	1.030e-01	20.060	0	Red	M
6	2840	6.500e-04	1.100e-01	16.980	0	Red	M
7	2637	7.300e-04	1.270e-01	17.220	0	Red	M
8	2600	4.000e-04	9.600e-02	17.400	0	Red	M
9	2650	6.900e-04	1.100e-01	17.450	0	Red	M
10	2700	1.800e-04	1.300e-01	16.050	0	Red	M
11	3600	2.900e-03	5.100e-01	10.690	1	Red	M
12	3129	1.220e-02	3.761e-01	11.790	1	Red	M
13	3134	4.000e-04	1.960e-01	13.210	1	Red	M
14	3628	5.500e-03	3.930e-01	10.480	1	Red	M

Summary:

```
  Temperature       Luminosity           Radius             Magnitude       Type             Color       Class
Min.   : 1939   Min.   :     0.0   Min.   :   0.0084   Min.   :-11.920   0:40   Red         :112   A: 19
1st Qu.: 3344   1st Qu.:     0.0   1st Qu.:   0.1027   1st Qu.: -6.232   1:40   Blue        : 55   B: 46
Median : 5776   Median :     0.1   Median :   0.7625   Median :  8.313   2:40   Blue-white  : 26   F: 17
Mean   :10497   Mean   :107188.4   Mean   : 237.1578   Mean   :  4.382   3:40   Blue White  : 10   G:  1
3rd Qu.:15056   3rd Qu.:198050.0   3rd Qu.:  42.7500   3rd Qu.: 13.697   4:40   yellow-white:  8   K:  6
Max.   :40000   Max.   :849420.0   Max.   :1948.5000   Max.   : 20.060   5:40   White       :  7   M:111
                                                                                 (Other)     : 22   O: 40
```

Since features like temperature, luminosity, radius and magnitude have different units, they have to be scaled to same level in order to get accurate results. Otherwise feature with bigger numbers will make bigger impact on the model results.

Summary after scaling:

Temperature	Luminosity	Radius	Magnitude	Type	Color	Class
Min. :-0.8959	Min. :-0.5974	Min. :-0.4586	Min. :-1.5478	0:40	Red :112	A: 19
1st Qu.:-0.7488	1st Qu.:-0.5974	1st Qu.:-0.4584	1st Qu.:-1.0078	1:40	Blue : 55	B: 46
Median :-0.4943	Median :-0.5974	Median :-0.4571	Median : 0.3732	2:40	Blue-white : 26	F: 17
Mean : 0.0000	Mean : 0.0000	Mean : 0.0000	Mean : 0.0000	3:40	Blue White : 10	G: 1
3rd Qu.: 0.4772	3rd Qu.: 0.5064	3rd Qu.:-0.3759	3rd Qu.: 0.8844	4:40	yellow-white: 8	K: 6
Max. : 3.0885	Max. : 4.1366	Max. : 3.3091	Max. : 1.4885	5:40	White : 7	M:111
					(Other) : 22	O: 40

After analysing this we can figure out mean and median of the numerical features don't have that much difference. So, we can conclude that heavy outliers aren't present in the dataset.

Visualization in Power BI:

Bar Chart:

Bar chart here shows the how each feature affects the type of the star.

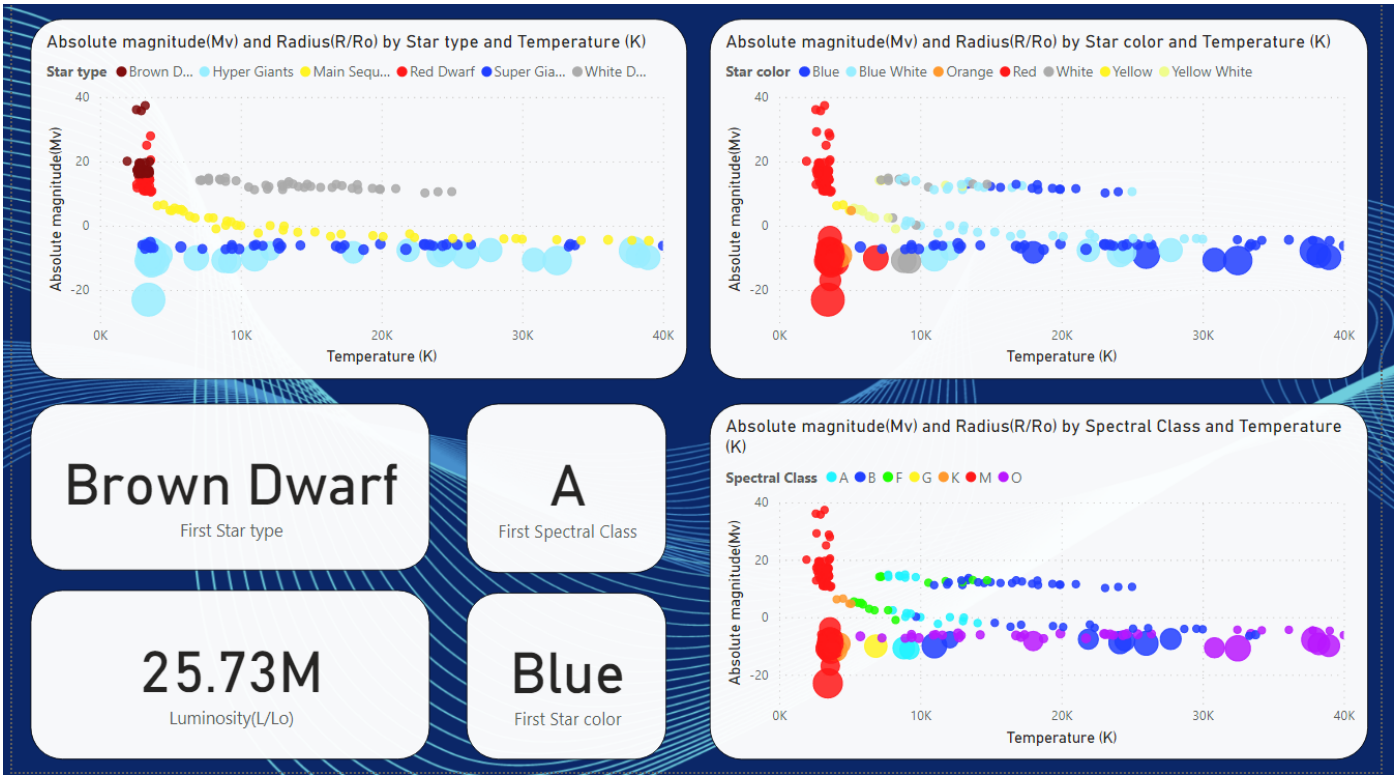
The absolute magnitude of a star is defined as the magnitude (brightness) it would have if it were viewed at a standard distance of 10 parsecs (32.6 light-years). From 1st graph we can understand that brown dwarf has the highest magnitude (positive side) and hyper giants has the lowest (negative side).

Luminosity is the amount of energy (light) that a star emits from its surface. 2nd graph shows hyper giants and super giants are most energetic ones and then luminosity significantly goes down when it comes to main sequence and others.

Radius indicates the size of the star. And 3rd graph shows hyper giants are largest. Their sheer size makes others look smaller in comparison. So it shows less significance when comparing other types.

When it comes to temperature graph shows that main sequence are the hottest. Main sequence are the middle aged stars like our Sun. Ones with lowest temperatures are red dwarfs and brown dwarfs. They are actually dead stars – last form of a star after super giants stage.

Scatter Plot:



With scatter plot we can actually compare it to the real HR diagram mentioned earlier. Temperature – Magnitude combination is the one which showed most similar looking diagram to the original HR diagram. Temperature at X-axis and Absolute magnitude at Y-axis. 1st one is classified based on star type, 2nd on the colour and 3rd on spectral class. And 1st one is the one we have to look into since type is our target class.

Hyper and super giants -> has negative magnitude from 0 to -20 and temperature ranges up to 40 K.

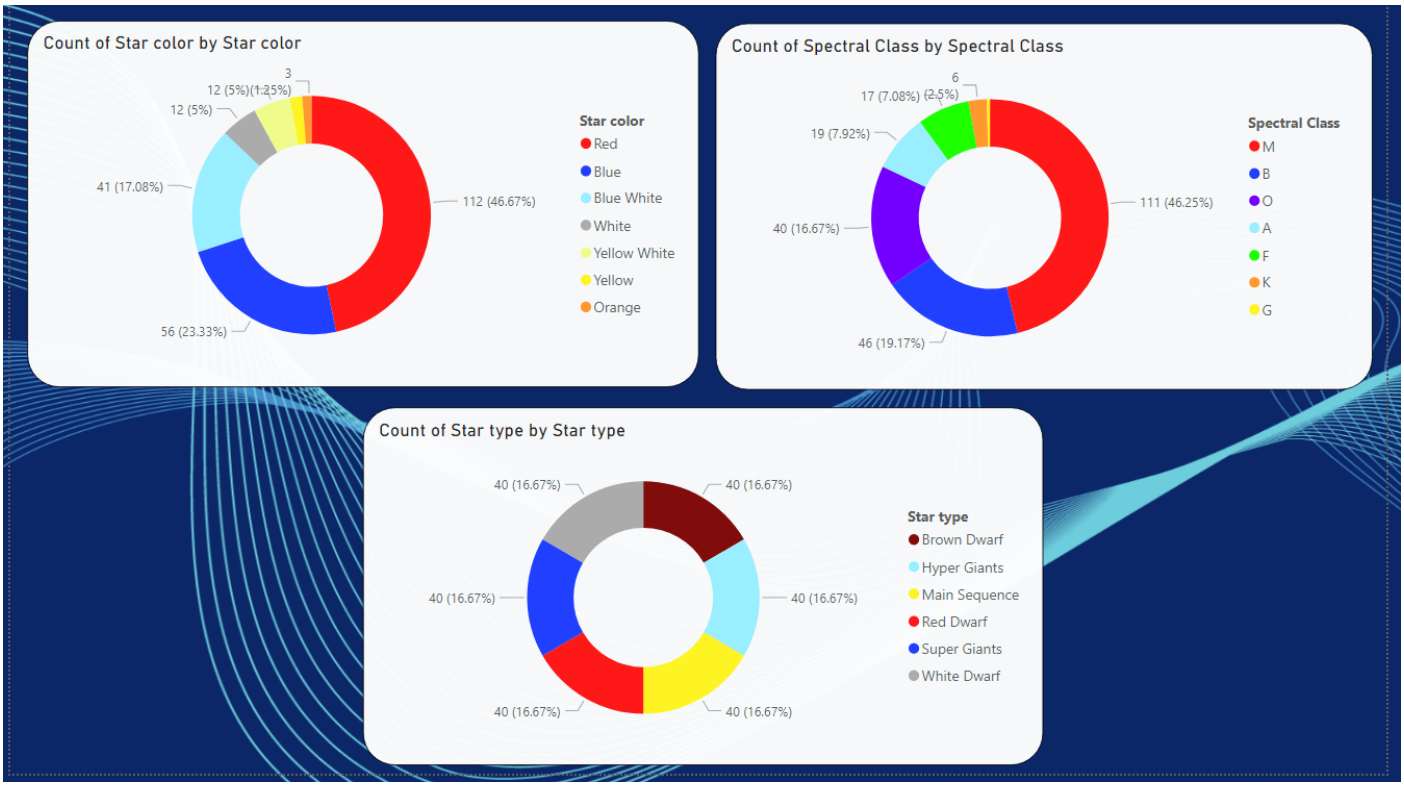
Main sequence -> magnitude ranges from 0 to 10 and temperature ranges up to 40 K.

White dwarf -> magnitude is around 20 and temperature ranges from 5 K to 30K.

Red dwarf -> magnitude ranges from 10 to 30 and temperature ranges around 3K.

Brown dwarf -> magnitude ranges from 20 to 40 and temperature ranges around 3K.

Pie Chart:



3rd chart implies that a sample of 40 stars from each type were taken for the dataset. That's why they are equally distributed.

1st chart shows majority of star in our universe is red coloured (around 46%) then blue (23%) and blue white (17%).

Similarly 2nd chart shows that majority of the star belong to M class (46%), then B class (19%) and O class (16%).

Model Building

Coding part is done with R language and the IDE used is R Studio.

Libraries used are:

library(caTools) *# for splitting data*

library(caret) *# classification and regression training*

library(e1071) *# for svm, naïve bayes*

library(class) *# for knn*

library(rpart) *# for tree*

Data is split into train and test in the ratio of 65% to 35%

1) KNN

model_knn <- knn(train = train[1:4], test = test[1:4], cl = train$Type, k = 3)

cm_knn <- confusionMatrix(test$Type, model_knn)

cm_knn

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1  2  3  4  5
         0 17  0  0  0  0  0
         1  0 17  0  0  0  0
         2  0  0 18  0  0  0
         3  0  0  0 17  0  0
         4  0  0  0  1 17  0
         5  0  0  0  0  0 16

Overall Statistics

               Accuracy : 0.9903
                 95% CI : (0.9471, 0.9998)
    No Information Rate : 0.1748
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9883

 Mcnemar's Test P-Value : NA
```

2) Naïve Bayes

model_nb <- naiveBayes(Type ~ ., data = train)

pred_nb <- predict(model_nb, newdata = test)

cm_nb <- confusionMatrix(test$Type, pred_nb)

cm_nb

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1  2  3  4  5
         0 15  2  0  0  0  0
         1  0 17  0  0  0  0
         2  0  0 18  0  0  0
         3  0  0  0 15  2  0
         4  0  0  0  0 18  0
         5  0  0  0  0  0 16

Overall Statistics

               Accuracy : 0.9612
                 95% CI : (0.9035, 0.9893)
    No Information Rate : 0.1942
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9534

 Mcnemar's Test P-Value : NA
```

3) Decision Tree

model_tree <- rpart(Type~., train, method = "class")

pred_tree <- predict(model_tree, test, type = "class")

cm_tree <- confusionMatrix(test$Type, pred_tree)

cm_tree

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1  2  3  4  5
         0 16  1  0  0  0  0
         1  0 17  0  0  0  0
         2  0  0 16  1  0  0
         3  0  0  0 16  1  0
         4  0  0  0  0 17  0
         5  0  0  0  0  0 17

Overall Statistics

               Accuracy : 0.9706
                 95% CI : (0.9164, 0.9939)
    No Information Rate : 0.1765
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9647

 Mcnemar's Test P-Value : NA
```

4) SVM

model_svm = svm(formula = Type ~ ., data = train, type = 'C-classification', kernel = 'linear')

pred_svm = predict(model_svm, newdata = test[-5])

cm_svm = confusionMatrix(test[, 5], pred_svm)

cm_svm

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1  2  3  4  5
         0 17  0  0  0  0  0
         1  0 17  0  0  0  0
         2  0  0 18  0  0  0
         3  0  0  0 14  3  0
         4  0  0  0  0 18  0
         5  0  0  0  0  0 16

Overall Statistics

               Accuracy : 0.9709
                 95% CI : (0.9172, 0.994)
    No Information Rate : 0.2039
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.965

 Mcnemar's Test P-Value : NA
```

Model Evaluation:

	KNN	Naïve Bayes	Tree	SVM
Correctly Classified	102	99	100	100
Misclassified	1	4	3	3
Accuracy	99.03	96.12	97.06	97.09

Conclusion

All the models are having high enough accuracy. But KNN proved to be the most accurate one in this particular problem with an accuracy of 99.03%. Because our dataset only had 240 rows and had no missing values and serious outliers. And numerical features were scaled accordingly. These factors gave KNN an edge over other algorithms.

Tree model's accuracy was switching between 97% and 100%. This inconsistency makes it less reliable for this problem. While KNN was consistently predicting with an accuracy of 99%.

Naïve Bayes works by assuming that features are independent while in reality our features have some kind of dependencies. Example: both magnitude and luminosity are different types of measurements of brightness. So they will show some dependencies. Because of this naïve bayes is also not reliable.

SVM is only good when the dataset have large number of features and small number of instances. Since our dataset is opposite to that, SVM is not a reliable one here.

So final conclusion is that we should go with KNN algorithm which is most accurate as well as the most reliable one too.

From an application point of view, this method will be highly useful for the astronomers who want to classify the newly found stars in the galaxies. This will be helpful for them to identify their age. Because each stage or type represents its age. By this they can easily figure out whether this particular star is new born or about to explode or dying.