Prediction of star type using machine learning algorithm K-Nearest Neighbour

Fadhluddin bin Sahlan (1817445)

CSC 3304 – Machine Learning (Section 1)

Date of submission: 7th December 2020

Prediction of star type using machine learning algorithm K-Nearest Neighbour

Stars are massive in space bodies that are made of various components. Most of the stars are made of hydrogen and helium and the reaction between these components produce their brightness and hotness. However, the distance between the earth and the stars other than the sun is light-years away making the brightness and the high temperature of the stars not affect humans. Only in the Milky Way galaxy there are an estimated 300 billion stars but there are only 2000-2500 that are visible as dots in the sky. There are several star types where it is measured based on several features of the star.

This paper is to build a predictive model that can classify star types based on features of the stars by using the K-Nearest Neighbour algorithm (KNN). Using machine learning in predicting the star type can ease people's life and also increase the efficiency of assigning type of new stars. It is known that there are many new stars discovered every year. In the Milky Way only there are seven new stars every year and every star will go through phase by phase and its type also changes by time. Being able to predict the stars automatically will be very useful for scientists.

From the dataset that has been retrieved from an open source dataset, some preprocessing data activity needs to be done in order to produce a processable data. The data now contains a lot of noise and needs to be cleaned and transformed into good data in order to use it as training data in the KNN model. This is because good data will produce a high performance model that will help scientists in determining the star types in a high accuracy manner. The work done will contribute to the open source world with the result of building models using the KNN algorithm. Others might use this paper to create other models with different algorithms using the same data and compare it with the current work.

Methodology

A dataset of stars with several features like temperature, luminosity, radius, absolute magnitude, star color, spectral class and star type is given. The dataset appears to have 240 rows of data and there are no null values appeared in any cell of the dataset. The dataset needs to be taken by the python program to be transferred into processable data.

Data Preprocessing

The first process is to do data preprocessing. The data needs to be cleaned up if there contains any null values or any outliers which in this case there are no null values and outliers. So, proceed to the next process of data transformation. In this process the dataset will be transformed into processable data. Columns like star color need to be updated to another variable that can be calculated which is floating point value. The problem here is this column contains inconsistent strings as there are cells that have the same color logically but different in their string value. The python program preprocess.py needs to take that matter into account in order to create a good dataset. The column 'star color' changed into a floating point value based on its respective color which is in the category of blue, blue-white, yellow-white, white and red where the values are 0, 1, 2 , 3, 4 and 5 respectively. The reason to convert the color to be represented as floating point is to make the column processable as string is a data type that is difficult to be processed. The next step is to convert all the rows to become floating points to ease the process of building the model. When the data is retrieved from a csv file, the data is in strings format which is difficult to be processed. The last step of data transformation is to change the variables that are not target variables to z-scale. The formula of z-scale is as below where μ is mean, x is the value of the cell and σ is standard deviation of that particular column:

$$z - value = \frac{\mu - x}{\sigma}$$

The reason for transforming the value of the cell to z-scale is to make every column have the same range of values and this is to prevent any column from dominating and discriminating against other features of the dataset.

The next step is features selection. Selecting the correct features will be very useful to create a high accuracy model. Choosing all the features might cause the dataset to be noisy as some features might not influence in determining the type of the stars. From some literature work on how scientists determine the type of stars, they classify them based on the temperature of the star in increasing order. There are also other factors that influence the type of stars such as luminosity, radius and color. From the dataset, it has features of temperature, luminosity, radius, absolute magnitude, star type which will be the target variable, star color and spectral class. From the literature work that has been done, absolute magnitude and spectral class do not affect the type of star thus these two features will be discarded from the dataset leaving the remaining features such as temperature, luminosity, radius, color and star type as the target variable.

Building the Model: K-Nearest Neighbours Algorithm

K-Nearest Neighbours is one of the machine learning algorithms in solving classification and regression problems. It is a distance-based algorithm in which each particular object or rows will be calculated its distance with the other rows. This algorithm is suitable to be used as a model for predicting star type as it is a classification process. There are many options to measure distance between objects such as manhattan distance and euclidean distance but in this paper, euclidean distance is used to measure the distance between objects. Below is the formula of euclidean distance:

$$d(p,\ q)\ =\ \sqrt{\sum_{i=1}^{n}(q_i\ -\ p_i)^2}$$

Before the building model process, the dataset will be divided into training data and testing data with the portion of 80% and 20% respectively. But before splitting the dataset, it appears that the data is sorted based on the star type which is the target variable. If the splitting process is done right away, there will be bias in the model as testing data and training data might cover certain star types and some star types might not be able to be classified to the correct target thus this will reduce the performance of the model. So before splitting the data into training data and testing data, the dataset needs to be shuffled its rows to provide random characteristics in training data and to avoid any bias to the model.

The model will receive training data in the feature space and the k value which will be used during the classification. From the training data the model will map all the objects in the training data to feature space which will be useful during the object classification. The way this algorithm classifies an object is by knowing the k number of training objects with the closest distance to the test object. The majority target variable of the k closest neighbour will be the target variable value for the new object. The good approach is to select k as an odd value to avoid a draw vote between the neighbours. Figure 1 describes how the KNN algorithm works in feature space.

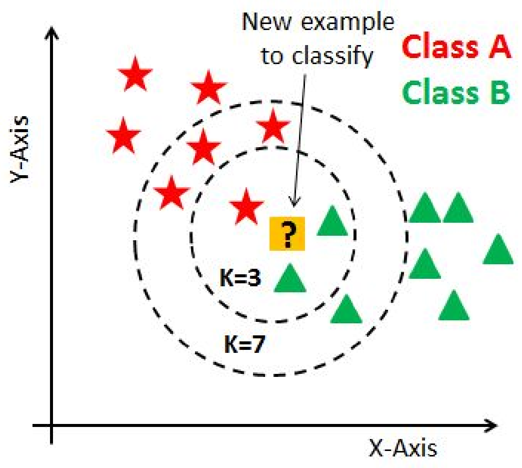


Fig. 1: KNN Algorithm in feature space

So the ‘**preprocess.py’** program will read raw data from the csv file and do all the preprocessing steps and produce two output files which are ‘Train_data.csv’ and ‘Test_data.csv that contain training data and testing data respectively. The ‘**model.py’** program will read data from both ‘Train_data.csv’ and ‘Test_data.csv’ file and build the model using KNN machine learning algorithm by using training data and predict the testing data then finally compare both the predict value and the actual value and perform some performance measurement. The program ‘model.py’ will be run multiple times with different k values and the performance measure will be recorded for each run.

Results

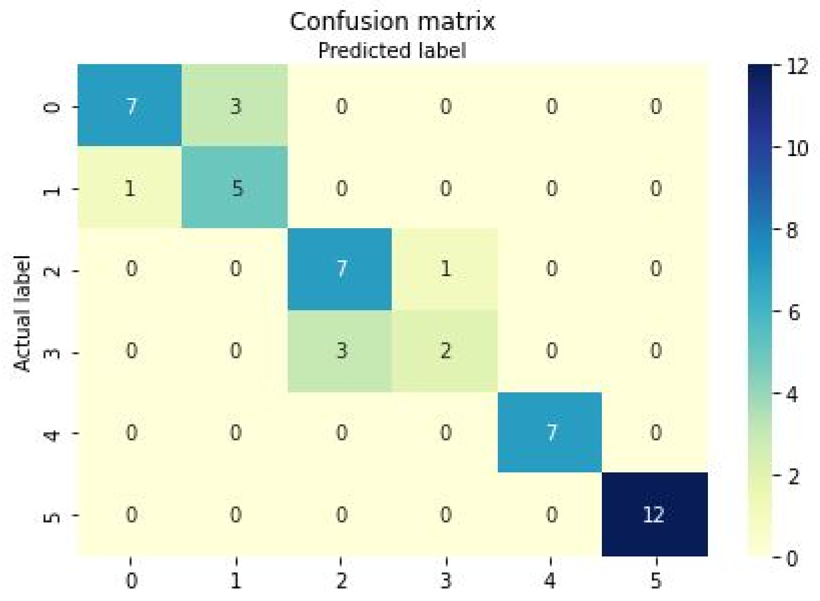
As stated previously, the model will be run multiple times with different k values. K values tested will be odd values from 3 to 11. The performance measure will be the confusion matrix and this will produce the measure of accuracy, precision and recall. Below is the result of the experiments.

<u>K = 3</u>

Actual : [5 4 4 4 3 4 3 5 2 5 2 1 0 5 2 1 5 2 5 0 0 0 4 1 1 1 3 3 0 0 1 3 2 4 0 2 0 5 5 5 5 0 2 2 0 5 5 4]

Preds : [5 4 4 4 3 4 3 5 2 5 2 0 0 5 2 1 5 3 5 1 0 0 4 1 1 1 2 2 1 0 1 2 2 4 0 2 0 5 5 5 5 0 2 2 1 5 5 4]

Confusion Matrix



Accuracy: 0.8333333333333334

Precision: 0.8111111111111112

Recall: 0.8013888888888889

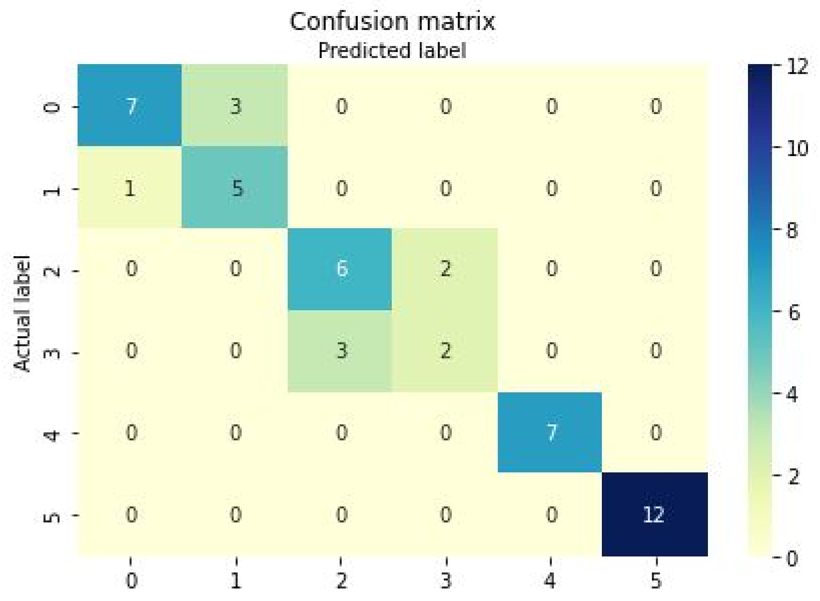
Using k value as 3 yields high performance of all three measures (accuracy, precision and recall) as all of them achieve a score more than 80%. From the confusion matrix, we can see that errors occurred close to the actual label.

K = 5

Actual : [5 4 4 4 3 4 3 5 2 5 2 1 0 5 2 1 5 2 5 0 0 0 4 1 1 1 3 3 0 0 1 3 2 4 0 2 0 5 5 5 5 0 2 2 0 5 5 4]

Preds : [5 4 4 4 3 4 3 5 2 5 3 0 0 5 2 1 5 3 5 0 0 0 4 1 1 1 2 2 1 0 1 2 2 4 1 2 0 5 5 5 5 0 2 2 1 5 5 4]

Confusion Matrix



Actual \ Predicted	0	1	2	3	4	5
0	7	3	0	0	0	0
1	1	5	0	0	0	0
2	0	0	6	2	0	0
3	0	0	3	2	0	0
4	0	0	0	0	7	0
5	0	0	0	0	0	12

Accuracy: 0.8125

Precision: 0.7777777777777777

Recall: 0.7805555555555556

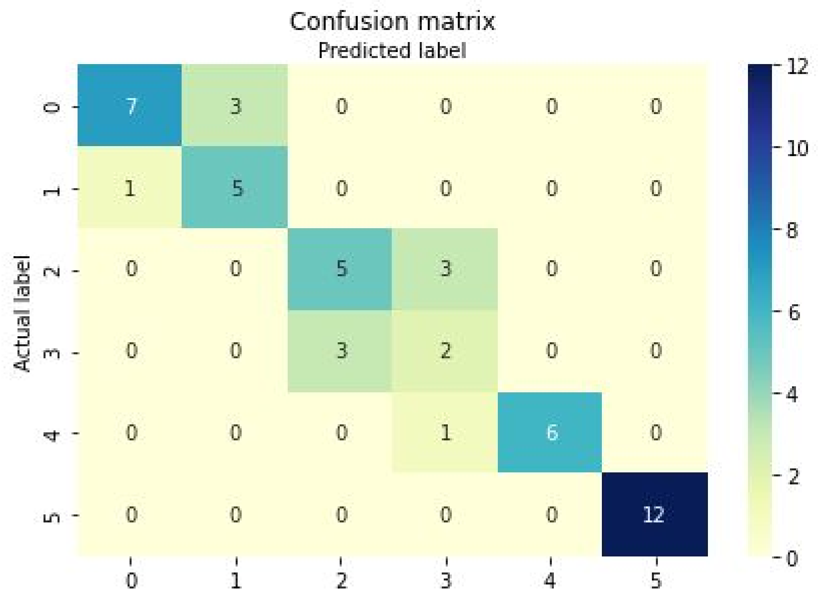
Increasing the k value to 5 results in an increase in error prediction by one as portrayed in the confusion matrix. This causes a slight drop in performance.

K = 7

Actual : [5 4 4 4 3 4 3 5 2 5 2 1 0 5 2 1 5 2 5 0 0 0 4 1 1 1 3 3 0 0 1 3 2 4 0 2 0 5 5 5 5 0 2 2 0 5 5 4]

Preds : [5 4 4 4 3 4 3 5 3 5 3 0 0 5 2 1 5 3 5 0 0 0 4 1 1 1 2 2 1 0 1 2 2 4 1 2 0 5 5 5 5 0 2 2 1 5 5 3]

Confusion Matrix



Accuracy: 0.7708333333333334

Precision: 0.7430555555555557

Recall: 0.7359126984126984

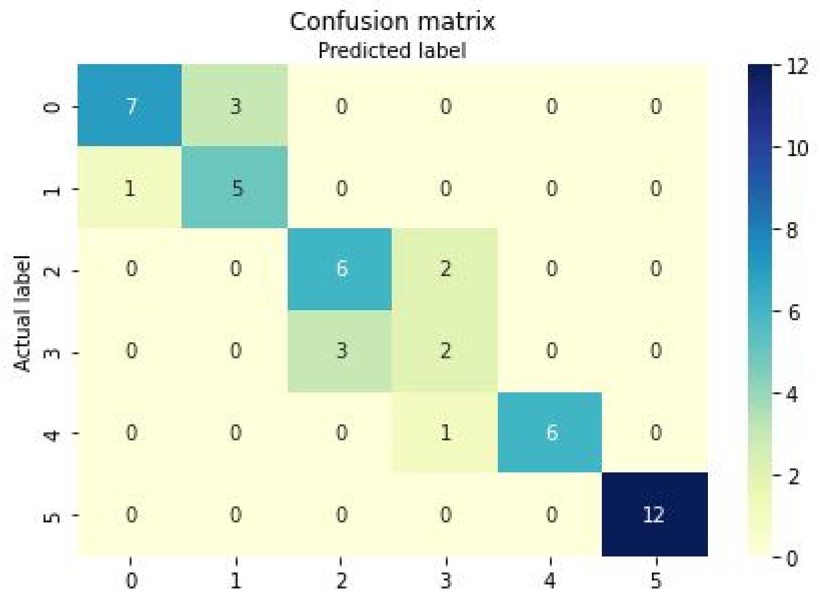
Further reduce the k value resulting in a drop of performance where the error assignment of object to the correct target increases to 2 as compared to when k value is 5. The pattern shows decrement of performance when the k value is increasing.

<u>K = 9</u>

Actual : [5 4 4 4 3 4 3 5 2 5 2 1 0 5 2 1 5 2 5 0 0 0 4 1 1 1 3 3 0 0 1 3 2 4 0 2 0 5 5 5 5 0 2 2 0 5 5 4]

Preds : [5 4 4 4 3 4 3 5 3 5 2 0 0 5 2 1 5 3 5 0 0 0 4 1 1 1 2 2 1 0 1 2 2 4 1 2 0 5 5 5 5 0 2 2 1 5 5 3]

Confusion Matrix



	0	1	2	3	4	5
0	7	3	0	0	0	0
1	1	5	0	0	0	0
2	0	0	6	2	0	0
3	0	0	3	2	0	0
4	0	0	0	1	6	0
5	0	0	0	0	0	12

Accuracy: 0.7916666666666666

Precision: 0.7611111111111111

Recall: 0.7567460317460317

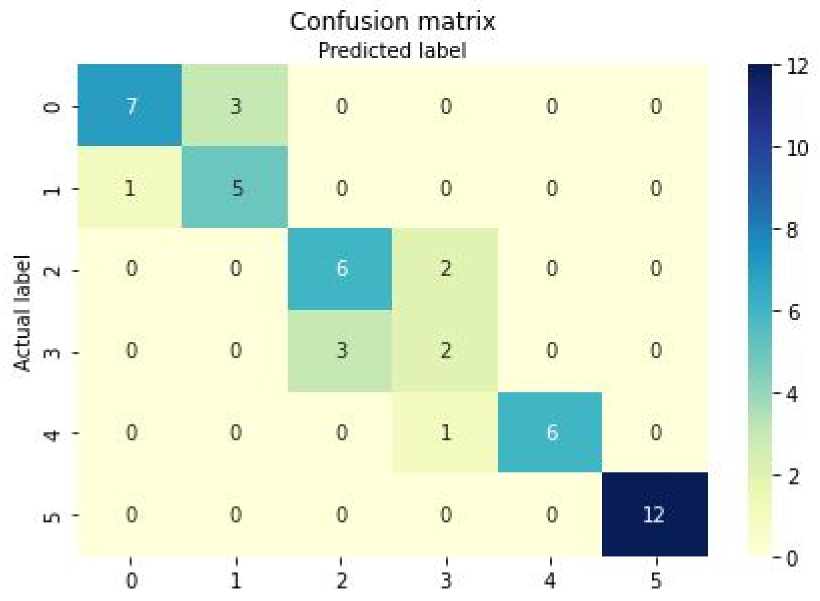
The k value further increases to 9. However the result portrays an increment in the performance thus changing the pattern that was already discovered before. This shows that decreasing k value does not mean increase the performance and vice versa.

<u>K = 11</u>

Actual : [5 4 4 4 3 4 3 5 2 5 2 1 0 5 2 1 5 2 5 0 0 0 4 1 1 1 3 3 0 0 1 3 2 4 0 2 0 5 5 5 5 0 2 2 0 5 5 4]

Preds : [5 4 4 4 3 4 3 5 3 5 2 0 0 5 2 1 5 3 5 0 0 0 4 1 1 1 2 2 1 0 1 2 2 4 1 2 0 5 5 5 5 0 2 2 1 5 5 3]

Confusion Matrix



Accuracy: 0.7916666666666666

Precision: 0.7611111111111111

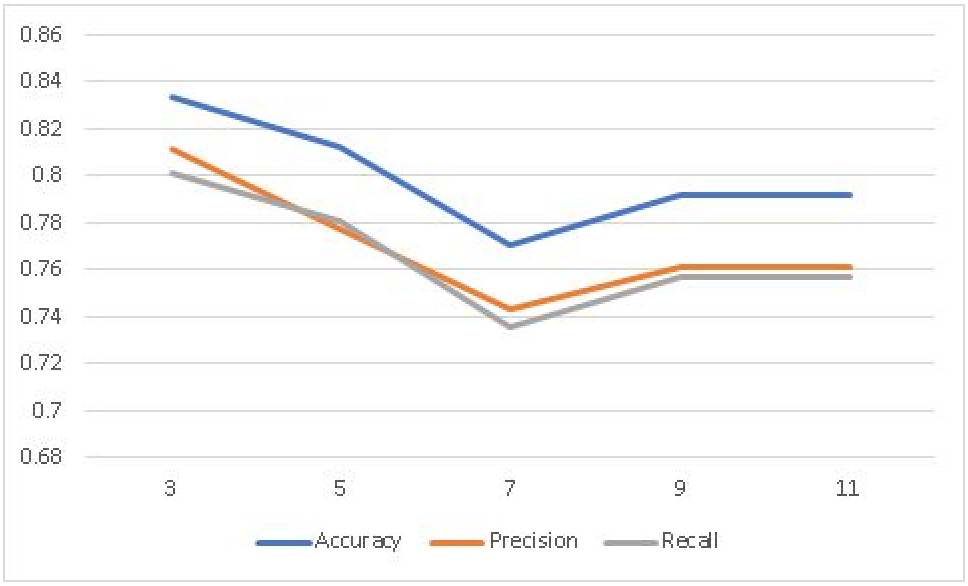
Recall: 0.7567460317460317

For the final run with the value k of 11, it shows the same performance as 9. From the result produced, it is obvious that KNN is a good algorithm for classifying star types as the highest accuracy, precision and recall achieved are 83.3%, 81.1% and 80.1% respectively. This is achieved when k is equal to 3. Further discussion on the results is in the next section.

Discussion

From the results obtained, it is approved that using KNN to build the model in predicting star types is one of a good choice. This is because the result showed high performance where when k = 3, the accuracy, precision and recall achieve more than 80%. Accuracy portrays the ability of the model to determine the correct label of the star's features. Recall shows the ability of the model to classify actual positive from labelling the object as positive. Lastly, precision tells the ability of the model to predict the true positive out of the positive values. The overall results of the KNN algorithm applied on predicting star types with various parameters' values of k is shown below in figure 2.

Fig 2: The performance of the model with different k values



Using k = 3 is the best solution in building a model that is able to determine the star type based on the features selected which all performance measures (accuracy, precision and recall) are peak at k=3. When the value of k increased to 5 the performance of the model portrays a drop as selecting the 5 neighbours of the test data might cause noise as it might also select data that are not in the same type as the test data. The model portrays further drop of performance when using 7 as the k value since it will only add more noise to the neighbour of the object that needs to be classified. When using 9 as the k value, the model shows an increment to the performance but not as good as 5 as the k value and selecting 9 neighbours might also select neighbours of the object that have further distance but with the same label. Further increase the k value shows convergence as when k is equal to 11 the performance remains the same but further increase the k value will affect the performance of the model. Choosing the right k value is important in building a good model and some iteration of building the model using different k values is needed in order to know which k is the best for the trained model.

The benefit of using this model is it can accurately predict the type of the unknown star provided with it the features needed. With the accuracy value is high it can predict the true positive value accurately out of all the values predicted. With the high recall value, the model can predict the actual positive accurately from all the actual positive values. While with the high precision value, the model can predict the actual positive precisely out of those predicted positive. However when using this model, it is not quite efficient actually in terms of speed. Increasing the k values means increasing the time to compute the result. There are other algorithms that are better in terms of computation time as compared to this algorithm.

Conclusion

In conclusion, KNN is a good supervised machine learning algorithm in solving classification problems such as predicting star types. It focuses on determining the type of the object by viewing the majority type of the object's neighbours in the features space. The results prove that KNN is a good algorithm in solving this problem since it achieves more than 80% in all measures of performance when k is equal to 3. Choosing the right k is important in yielding a good model as it affects the performance of the model in classifying things. Further enhancement of this paper is, it is better to split the data based on the target variable for example if the split is 80% training data and 20% testing data, it is good to split 80% of type 0 to training data and 20% of type 0 to testing data and there go the rest of target variable. This is to prevent bias in the data and to avoid discrimination a specific dependent variable in training the model and also to make sure the model is trained with all types. The training data will need to be shuffled after the split to provide randomness in training data.