Capstone Project

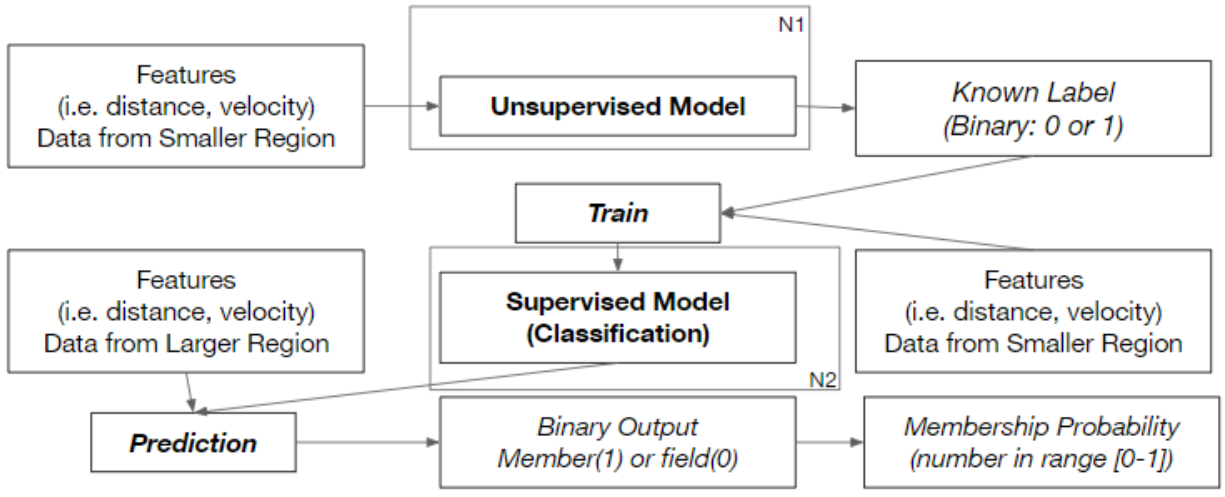
A Semi-supervised Machine Learning Approach for Open Star Cluster Member Determination

Md Mahmudunnobe

Minerva Schools at KGI

Executive Summary

When a group of stars is born together from the same cloud of gases, it is called a star cluster. Analysis and research on star clusters help us understand star formation and how they evolve. But first, we need to know which stars are members of a cluster before analyzing them. Most of the available methods for member determination used an unsupervised approach, where we divided the stars into two or more separate groups based on their properties. The supervised methods are usually better in terms of efficiency and validation, but we need labeled training data. As shown in the flowchart, I propose using an unsupervised method to generate a training dataset. Then we can use that dataset to train a better supervised model, which will allow us to detect more reliable members.



The flowchart of my project showing how I am planning to combine both unsupervised and supervised methods in a semi-supervised approach.

Table of contents

Acknowledgement

I would like to acknowledge and give my warmest thanks to my advisors Dr. Ben Richard and Dr. Raquel Ribeiro who provided constant support and important feedback throughout the year. Their guidance and advice helped me develop a strong project. I would also like to thank Dr. Priya Hasan and Dr. Najam Hasan from Maulana Azad National Urdu University in Hyderabad, India for their feedback on the astronomy related topics. I want to give a massive thanks to Dr. Rohan Shekhar, Dr. Kiel Howe, and Dr. Erin Kamler for guiding me during the primary stage of my project. I want to thank my peers including Albion, César, Jatin and Isabelle for providing a helping hand in all things from accountability to moral support throughout this process. Finally, and most importantly, I thank my parents and relatives for their help in all my academic and personal endeavors.

Abstract

The members of an open star cluster are often hard to distinguish from the field star in the same area of the sky, and various machine learning (ML) methods have been used to solve the problem. This paper proposes a semi-supervised ML model, where we can use an unsupervised model to generate training data and then use that to train a supervised model. First, I justified the choice of evaluation metrics that can be used to compare models to find the best one. As an example, I applied Gaussian Mixture Model and Random Forest for M67 and a new cluster NGC 3766. I discussed the best practices to justify the decisions while running a model, including feature selection and model hyperparameter optimizations. Lastly, I conclude by laying out the future directions.

Keywords: Astrophysics, Machine learning (ML), Open Star Cluster (OSC), Semi-supervised algorithm

1. Introduction

Star clusters are an important field of study in astrophysics. As many stars are born in a star cluster, the study of the star clusters helps us to understand the formation and evolution of stars and stellar bodies. But before any analysis, we first need to know the reliable members of a cluster. In this paper, I proposed a semi-supervised approach to detect members from the observed archival data.

In section 2, I discussed the necessary background topics, and in section 3, I explored the description and justification of my hypothesis as well as the workflow of the process. In the next two sections, I applied two example machine learning models. As an unsupervised model, Gaussian Mixture Model is studied and applied in two clusters. Then in the next section, I go over Random Forest as an example of a supervised model. Then, after a brief result section, we conclude with the future work in section 7.

2. Background

2.1. Star Cluster

When a group of stars originates from the same interstellar cloud at the same time and are held together lightly by mutual gravity, the group of stars is called a star cluster. Almost all the stars of a specific cluster have the same age, the same distance from us, the same relative motion compared to us, and the same chemical compositions (Platais, 2009). But when we observe the very distant star clusters, it becomes difficult to distinguish the members from the other stars projected in the same direction of the sky, either in the foreground (closer than the cluster) or background (beyond the cluster) of the cluster stars (Kharchenko, 2005). Many different analytical methods have been developed and still developing to improve the accuracy of finding the membership probability (how likely it is to be a member) for a given star in a large field of stars containing a star cluster (Stott, 2018). Along with other branches of astronomy, this membership detection process also started to use different machine learning approaches since the beginning of the 21st century.

The membership detection approaches can be broadly classified into two groups based on the type of input variables: astrometric and photometric approaches.

2.2. Astrometric Approach

Astrometric variables are the ones that we can detect and quantify from an image of the stars. This includes the position of the star in the sky plane (i.e., right ascension, RA, and declination, dec), its relative translational motion over time (proper motion), the shift of its position as the earth moves around the sun (parallax) which is inversely proportional to its distance. These variables are densely distributed for members of a cluster compared to the random stars in a projected sky area, thus very helpful to use in cluster membership. But one

problem is, for distant clusters, the position shifts are too small to detect any proper motion or parallax with most of the instruments.

The basic idea for all astrometric approaches is quite simple. Use a model to find the group of stars that have similar RA, dec, parallax, and proper motion in *ra* and *dec* compared to the other field stars. Among them, the most important and most used variable for cluster membership is the proper motion (pm: Balaguer-Núñez, 1999; Yadav et al., 2013).

We will use the following astrometric variables in our analysis in this paper (the units of these variables are given in square bracket):

1. *ra*: right ascension or the horizontal angular position of the stars in the sky [degree, deg]
2. *dec*: declination or the vertical angular position of the stars in the sky [degree, deg]
3. *pmra*: proper motion along the right ascension direction of the sky plane [milli arcsec per year, mas/yr]
4. *pmdec*: proper motion along the declination direction of the sky plane [mas/yr]
5. *parallax*: the shifted angle of the star as the earth moves around the sun. It is used to measure the distance (inversely proportional to distance). [mas]

2.3. Photometric Approach

Photometric variables are the ones that come from the photometric measurements: where we measure the intensity of light from the star in different band filters (i.e., blue, visible, infrared, etc.). This includes primarily the flux and the apparent magnitude in different bands and the difference of magnitude between two bands (color index or simply ‘color’).

If we plot the luminosity (energy emitted per second) and the temperature of different stars, it follows a very distinct plot called the “HR diagram”. Similarly, if we have a group of stars of similar age but varying mass (just like a stellar cluster), we will find them scattered along

this HR diagram. Luminosity is related to magnitude, and temperature is related to color, so we can express the HR diagram as a color-magnitude diagram. So, after observing photometric variables, if we plot a color-magnitude diagram, the members of a cluster should lie along an H-R diagram, where the field stars should have a random position.

Fig 2.1 shows a color-magnitude diagram, where we used absolute magnitude on the y-axis and the difference between the BP band and RP band magnitude (BP-RP color) on the x-axis. We can see that the top of the CMD is changing slightly as the age of the stars increases. On the right, we overplot a set of cluster members. Note that cluster members are not normally distributed in the color axis or in the magnitude axis, but they closely follow the theoretical CMD.

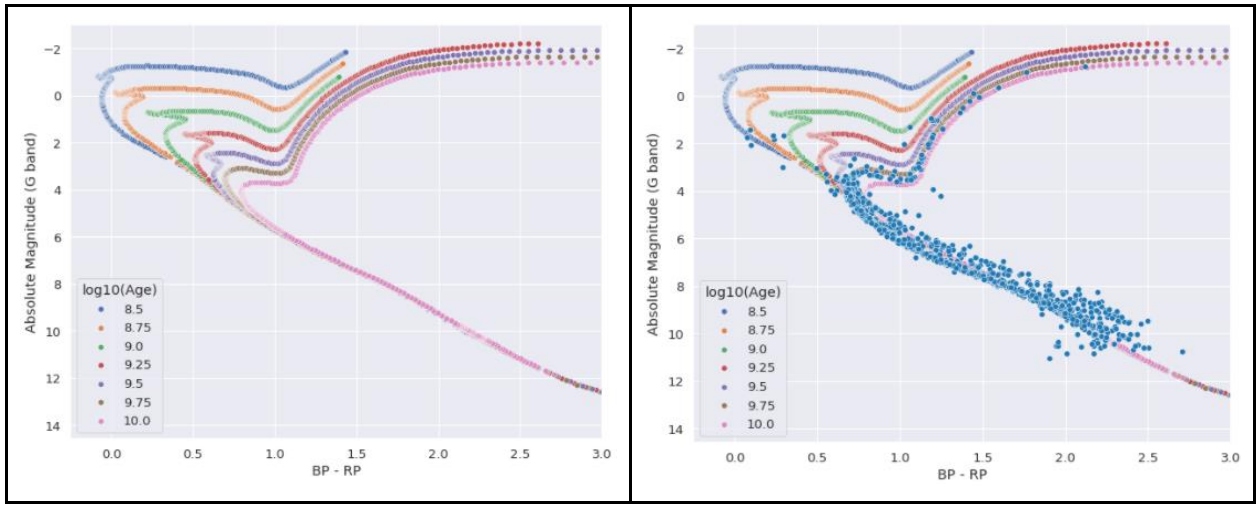


Figure 2.1: Theoretical HR or Color-Magnitude Diagram (CMD) for varying ages. The diagonal line is the main-sequence line and the somewhat horizontal line on top is the giant line. We can see as the age of the cluster increases, the main-sequence line becomes shorter and *turn-off* from main sequence to giant happens at higher magnitudes (fainter stars). On the right, a set of cluster members are plotted as blue dots, which closely follows the CMD. From its *turnoff point*, we can estimate that the age of cluster is around $10^{9.75}$ years.

The diagonal line of the CMD that goes from the bottom-left to the top-right is called the main sequence line. The main sequence star balances the inward pressure of gravity due to its mass by the outward pressure created by the nuclear fusion of hydrogen to helium at its core.

Very young stars who did not become the main-sequence star yet, usually lie just above the main sequence line on the bottom right corner.

The region on the top with the horizontal lines is called the supergiant region. Once a star finished burning all effective hydrogen in its core and started burning helium, it entered the giant phase. Stars of this phase are usually unstable, and their outward pressure for nuclear fusion is higher than the internal gravity, resulting in an expansion of the outer shell.

The region of the CMD where the main sequence line starts going towards the giant region is defined as the ‘turning point’. Depending on the mass of a star, its duration as the main sequence star (thus how long it will stay in the main sequence line) changes. In a star cluster, all the stars are of almost similar age but with different masses. Depending on the masses, some are still on the main sequence line, some have already gone to the giant region, and some just finished their main sequence duration and now turning towards the giant stage. So, we get a full distribution of stars over the CMD. For the stars that are currently on the turning point of CMD, their main sequence lifetime is equal to the age of the cluster. For all clusters, the lower part of the CMD, the fainter stars with larger magnitude thus fainter, usually becomes thicker, a

There is a relationship between the mass and brightness of a star. Brightness is simply a measurement of the total energy radiated from a star per second. The more massive a star is, the more its gravitational power and the more the surface gases are attracted to the core. In order to balance a larger inward pressure, the star needs a larger outward pressure, which requires burning more hydrogens in a unit of time. Luckily, the more inward pressure increases, the density and temperature of the core increase, which in turn increases the rate of nuclear fusion reaction. Due to this balancing feedback loop, a more massive star radiates more energy from its core and looks brighter in the sky.

As a more massive star has a higher nuclear reaction rate, it finishes burning all core hydrogen faster, thus having a short duration in the main sequence. So, if we can measure the brightness of the star at the turning point, we can find its main sequence lifetime, which will be the age of the star cluster. In Fig 2.1, we can see that as the age of a cluster changes, its turning point also changes. This is how photometric observations help to find the age of a cluster.

Similarly, if we know whether the cluster is younger or older, then we can estimate what the CMD of the member stars should look like. As shown in Fig 2.1, for a younger cluster (age of less than 100 million years (10^8 years)), most of the members should lie in a longer main-sequence line. The turn-off point will likely not be present, and only a few, if any, stars would be in the giant region. We can also see some additional pre-main-sequence stars above the main sequence line in the bottom right corner. For older clusters (100 million years or older), many of the members will go to their giant phase. Thus, the main sequence line will be shorter, and the turn-off point will be clearly visible.

One additional advantage of the photometric variables is that they are easier to observe precisely than a similar precise observation of the astrometric variables, but the limitation is that they are less useful to find the membership probability. As we can see in Fig 2.1, neither color nor magnitude is more normally distributed for cluster members compared to those of the field stars. It is more common to use the photometric variables to verify the members classified by the astrometric approach and remove any non-members as they all should lie in the HR diagram (Yen et al., 2017; Monteiro & Dias, 2019). In an only photometric approach, usually, we find the random color-magnitude diagram for the field star using a reference image where the cluster is not present. Then we subtract this from the color-magnitude diagram of the (field and cluster) image to find the member stars.

In this paper, when necessary, we will consider the following photometric variables:

1. *G:* G band Magnitude
2. *BP*: BP band Magnitude
3. *RP:* RP band Magnitude
4. *BP-RP:* the difference between BP and RP magnitude (or BP - RP color)
5. *BP-G:* the difference between BP and G magnitude (or BP - G color)
6. *G-RP:* the difference between G and RP magnitude (or G - RP color)

2.4. GAIA Mission

Recently European Space Agency (ESA) launched the *Gaia* (Global Astrometric Interferometer for Astrophysics) mission to measure astrometric data in about micro arcseconds accuracy (Perryman et al., 2001; Gaia Collaboration et al., 2018), which opened a series of works on open star clusters in recent years both to discover new open clusters (Sim et al., 2019; Liu & Pang 2019; Castro-Ginard et al., 2019, 2020) and new members in the open clusters (Cantat-Gaudin et al., 2018; Gao 2018a; Agarwal et al., 2020). In this project, I will use the 2nd data release (DR2) and the early 3rd data release (EDR3) of the Gaia mission, which is publicly accessible through astropy and astroquery packages in python.

2.5. Machine Learning Approach

The ML models used so far can be divided into two broad categories: parametric and non-parametric. In the parametric model, we model the output variable (membership probability) and the input variables (astrometric or photometric) using a function with a pre-determined form. For example, in linear regression, we assume that our output variable is a linear combination of the input variables, or in a Gaussian Mixture Model, we assume that our data is coming from a mixture of several normal distributions. Then using the observed data, we try to find the optimal

value of some specific parameters of the function (i.e., the coefficients of the linear function, mean, and variance of the normal distribution). These models are usually simpler to understand and interpret and take less time and fewer data to train, but if the real data do not follow the assumed form of the function, the model will provide a very poor fit (James et al., 2017). The common parametric methods include maximum likelihood (Sampedro & Alfaro, 2016; Jackson et al., 2020), Markov Chain Monte Carlo (MCMC) method (Hidayat et al., 2019, Bossini et al., 2019), Gaussian Mixture Model (GMM) (Gao, 2020; Agarwal et al., 2020), and Bayesian Inference using both astrometric and photometric variables (Maíz, 2019; Parren et al., 2015; Stott, 2018).

In a non-parametric approach, we do not assume any fixed form of the underlying mapping functions between the input and output variables. Instead, these models try to best fit the given data to construct the underlying function. So, they are flexible enough to fit different types of functions and do not need to make any strong assumptions about the given data. But usually, it needs a large amount of data, it is computationally expensive, and it can often overfit the training data (James et al., 2017). Some of the common non-parametric models are k-means clustering (El Aziz et al., 2016), Random Forest (RF) (Gao, 2018a), and Artificial Neural Network (ANN) (Castro-Ginard et al., 2018, 2019, 2020). Some people also used the 'Density-Based Spatial Clustering of Applications with Noise' (DBSCAN) algorithm (Gao, 2014; Bhattacharya et al., 2017, Castro-Ginard et al., 2018). Cantat-Gaudin et al. (2018, 2020) applied the UPMASK (Unsupervised photometric membership assignment in stellar clusters) algorithm, which is a combination of k-means clustering, principal component analysis (PCA), and kernel density estimations to Gaia data to identify members and derive their membership probabilities for 1481 open clusters.

Another important classification of these ML models can be done based on the necessity of training output data. The models that work without any labeled training data are unsupervised learning models. It is easier to use an unsupervised model as we do not have to collect or create training data. But in this model, it is hard to evaluate the model performance as we do not know the true output. The observational data of the stars do not give us any direct labels regarding the membership of a particular cluster; thus, it is more common to use an unsupervised model. Most of the above-mentioned models are unsupervised learning models, i.e., the Gaussian Mixture Model (GMM) (Gao, 2020; Agarwal et al., 2020), DBSCAN algorithm (Gao 2014; Bhattacharya et al., 2017), UPMASK (a combination of k-means clustering and principal component analysis, PCA) (Cantat-Gaudin et al., 2018, 2020).

The models that need training data with labeled target values are called the supervised learning models. Due to the presence of the true label, it is easier to validate the model performance as well as to train a more accurate model to map the input and output data. But it is not common to get the labels beforehand, and when possible, it is time and resource consuming. In this membership determination problem, only a few papers applied an additional supervised model (i.e., Random Forest (RF), Artificial Neural Network (ANN)) where they used the results of the unsupervised model as their training data. For example, Gao (2018a) used GMM and RF together, Castro-Ginard et al. (2018) used DBSCAN and ANN, and Mahmudunnobe et al. (2021) used a combination of UPMASK and RF. Gao (2018a) reported the discovery of an escaped member of the M67 cluster using their method, which was not found in the unsupervised model as it lies slightly further than the cluster radius. Mahmudunnobe et al. (2021) found 1.5-2 times more members for selected 9 clusters compared to the current open star catalog (Cantat-Gaudin et al., 2020)

One major limitation that we can see in all studies using the ML model is that they did not compare different models for their analysis. Most papers chose a certain ML model, reported their result and analysis using this model, and compared their retrieved member with previously found members. But for any given problem, there exist multiple different candidate ML models, and one way to justify our choice of model is to compare these candidate models. The presence of so many past studies with different ML models indicates that many ML model has the potential to distinguish member stars from the field star contamination. But we are still missing a comprehensive review study to compare different models, their robustness and performance, analyze their advantage and limitations and finally conclude which model (or set of models) are more appropriate for a given star cluster.

Another limitation is the lack of a common quantitative metric for evaluating the robustness of unsupervised models. In these studies, we found three main ways they tried to evaluate the performance of their chosen model. Firstly, the most common one is the visual inspection of the CMD for the retrieved members (Gao, 2014, 2018a; El Aziz et al., 2016; Bhattacharya et al., 2017; Stott, 2018; Agarwal et al., 2020). As explained in section 2.3, depending on the age of the cluster, it should show a longer or shorter main sequence line and the turn-off point. They justified that if their model is accurate enough to get the members, then the CMD of the members should very closely and tightly (i.e., with smaller variance) follow the theoretical CMD. Other visual metric includes the proper motion plot (*pmra* vs *pmdec*) (Agarwal et al., 2020) or parallax distribution (Jackson et al., 2020), where the members should stay very close to each other in a normal distribution, where field stars would be random and uniform.

Additionally, many of the studies (Stott, 2018; Cantat-Gaudin et al., 2018, 2020; Hidayat et al., 2019; Bossini et al., 2019, Agarwal et al., 2020) compared their result with the previous

studies to justify the robustness of their model. Some of them (for example, Stott, 2018) directly compares the number of the retrieved member with the past studies, whereas other (Agarwal et al., 2020) compared the members using the misclassification rate (number of predicted members (or field) that are confirmed field (or member) in past studies / total number of predicted member). A few studies (Cantat-Gaudin et al., 2018, 2020; Bossini et al., 2019) compared the value of different cluster properties (i.e., mean distance, age, etc.) derived from the retrieved members with the earlier studies.

The third and less common way to validate the unsupervised model was to apply it to simulated data (Sampedro & Alfaro, 2016; Parren et al., 2015). Using the true labels in the simulated data, they calculated the completeness (number of retrieved members by model/number of true members in the data) and misclassification rate.

The use of different metrics in different studies makes it very hard to compare their relative robustness. Visual metrics (CMD or proper motion distribution) are very subjective. Though they help us to distinguish between relatively good results and quite bad results, it is very hard to detect and quantify smaller increases or decreases in the performance of models using them. The comparison and simulation metric, for some studies, gave a quantifying number, but they require an additional component (a benchmark past study or a simulation) other than only the predicted member and field star group. The different studies used different benchmark past study (or studies) because as time progresses and new dataset arrives, new research comes with more complete and updated member information. Similarly, different authors used different assumptions while creating the simulated data. So, we cannot directly compare their metrics unless we can replicate their simulation properly with all their assumptions. Thus, we can see that in past literature, we are missing a common quantitative metric that can be used by any

unsupervised model with only its predicted member and field star groups to evaluate the model performance.

Therefore, we can summarize our findings from the literature review of the membership determination problem as follows. Due to the lack of precise astrometric measurement, most of the past studies focused more on photometric variables until the recent GAIA mission. After the first data release of GAIA in 2018, all studies incorporated astrometric variables in their analysis. Among the ML models, both parametric and non-parametric models are common. Most studies used an unsupervised model due to a lack of training data, though the addition of a supervised model has increased the retrieved member in a few studies. There is no study that compares two or more different ML models to choose the best one. Similarly, there is no common quantitative metric used in the literature to evaluate the performance of the unsupervised model. To fill these gaps, in this project, I will focus on incorporating supervised models as well as proposing a common quantitative metric to evaluate and compare different unsupervised models.

3. Project Description

3.1. Specific Focus: Hypothesis

As we discussed in the last section, machine learning literature suggests that supervised models are better to validate and get a more confident classification since we have a ground truth to evaluate the results (Jain et al., 1999, Reddy et al., 2018). But in real life, it is hard to get the labeled training data. Then we saw that some papers (Gao, 2018a; Castro-Ginard et al., 2018; Mahmudunnobe et al., 2021) used an unsupervised model first to get the training data and then fed that into a supervised model to get their final membership probabilities. All those papers reported an increase in members in this approach.

Based on this evidence, I argue that a well-justified and optimized semi-supervised model (combination of an unsupervised and a supervised model) would be a better approach compared to an unsupervised model to detect members of open star clusters in terms of both model performance and retrieved member. When applying an unsupervised model, to increase confidence in our model result, we need to apply it for a smaller search radius where field star contamination is small. The higher the search radius, the higher the field contamination (i.e., field-member ratio), and the lower the confidence. This approach often results in a smaller group of retrieved members. In a semi-supervised model, we can use this smaller but reliable member group as our training data and apply an additional layer of a supervised model. As we now have training data, we can explore further in the feature space (for example, stars in a larger search area), which can allow us to find stars that are very similar to the training stars. This is how we should expect to get a larger number of member stars while still having strong confidence in our results from the semi-supervised model.

I need to define what I meant by a well-justified and optimized semi-supervised model. While running an ML model, we always need to make several different choices, including what features to use, how to process the data, etc. Similarly, most if not all ML models have some hyperparameters that we need to choose beforehand, and that affects the model performance. For example, K-Nearest Neighbor (K-NN) is a supervised model which classifies a given point by comparing its k number of nearest points in the training data. Here, k is a hyperparameter that we need to choose earlier. A well-justified and optimized model is the model where all the different choices that we made are justified properly, and the hyperparameters are chosen in a way that optimizes the model performance. Additionally, for such a semi-supervised model, we also need to justify which supervised and unsupervised model we want to use.

In the mechanism of the hypothesis, I presented a logical argument with the following four premises and a conclusion. If we want to be more confident in the unsupervised model, then we need to decrease the field star contamination (1. if A, then B). If we want to decrease the field contamination, then we need to lower our search radius (2. if B, then C). If we lower our search radius, then we retrieved a smaller number of members (3. if C, then D). We want to increase our model confidence (4. A). Therefore, we will have a smaller number of members (therefore, D). Here the conclusion directly follows the premises with formal logic. Thus, this is a valid argument.

Now to check its soundness (i.e., the truth value of the premises), we need to understand how an unsupervised model works in this context. In an unsupervised model, we tried to group the dataset (which consists of normally distributed members and randomly distributed field stars) into two different groups. Unlike the supervised model, we do not have any prior knowledge about the parameters of these two groups. When the number of random field stars increases,

more of them overlap with the members, and it becomes harder for the model to distinguish the member and field efficiently. Thus, for many unsupervised models, the confidence could be inversely correlated with the field star contamination. We cannot say that this is true all the time, but it is highly plausible. The second and third premises are true most of the time. When we search around the cluster center, we can expect that most of the members will be close to the center. For a wider search area, further regions should have more field stars and only a few members. Thus, one way to remove field contamination is to narrow our search radius. When we do that, we also lose all the members that are a bit further from the cluster center.

Another assumption in our mechanism is that for a supervised model, the confidence does not depend on the field star contamination. This is also very plausible because we already have training data, and we can map the parameters of the stars with the training data to determine their group (member or field). As long as we have reliable training data, we can still get good confidence for a dataset with large contamination.

To test the hypothesis, first, we need to build the semi-supervised model. There are two important considerations in this regard. First, we need to clarify what will be the structure of this model and how do we plan to combine the unsupervised and supervised models together. This will be discussed in detail in the following subsection (section 3.2). Second is the choice of supervised and unsupervised models. To find the best unsupervised model, I will compare several candidate unsupervised models and choose the best one. As discussed in section 2.5, from the literature, we do not have any common metric for comparing unsupervised models in this context. Thus, I will propose a metric that can be used to quantify the performance of any unsupervised model for membership determination. Similar to the unsupervised model, I will

also compare several candidates supervised models using the chosen performance metric and select the best one. The evaluation metrics for both models will be discussed in section 3.3.

One important point to keep in mind is that before comparing any model, we need to make sure that they are properly well-justified and optimized. For the limit of this project, I chose Gaussian Mixture Model (GMM) and Random Forest (RF) as an example of the unsupervised and supervised models, respectively. In the following two sections (sections 4 and 5), I will thoroughly discuss how we can make these models well-justified and optimized. Ideally, this is what I want to do for all possible candidate models. Once they all are well-justified and optimized, then I can compare the unsupervised models to select the best one and, similarly, the best supervised one. Finally, we can combine them together in a semi-supervised model as explained in section 3.2 to get our final model.

Once we have our semi-supervised model, to be able to test the hypothesis, we need to compare two outcomes: the number of retrieved members and model performance. For the first one, we can just find the recent studies with the unsupervised model(s) from the literature and compare the number of members with our result. The best choice from the literature will be the Cantat-Gaudin et al. (2020) paper, which is the current benchmark for the open star cluster catalog. This paper used the 2nd data release of the GAIA mission and provided the predicted members for all identified open star clusters in our galaxy. They applied the unsupervised model UPMASK (a combination of k-means clustering and PCA) in their analysis.

Now to compare the model performance, we need a common metric that can be used for both our model and the unsupervised model(s) used in the literature. As discussed earlier, there are mainly three metrics found in the past studies: visual inspection, comparison, and simulation.

We can make a visual inspection to see how the CMD or the proper motion distribution changes between our model and the literature model. It may inform us about the regions where we have more (or less) members, but it will not provide any quantitative measure of the model performance. Regarding comparison metric, we could only choose the paper which gives a quantitative measurement (for example, misclassification rate like Agarwal et al., 2020) with respect to past studies. But here, the limitation is that we are assuming that the past studies chosen in that paper are ground truth and do not have any false positive or negative. Lastly, if we chose a paper that used simulation for their evaluation, we need to know all their assumptions behind the simulation and be able to replicate the simulation efficiently. But still, the limitation will be the assumption that the simulation reflects the real data properly.

A better approach could be to use a metric that can evaluate the model performance only using its predicted member and field star group. Every past paper divides the data into two groups: member and field. So, in principle, it is possible to test their performance using the new metric. We can again use the current benchmark Cantat-Gaudin et al. (2020) paper for comparing the model performance. But most of the paper including this only reports the member dataset with their paper, as the field dataset is not useful for any further analysis of the cluster. So, in practice, we need to replicate their unsupervised model: apply it to the same dataset, following the same procedures and taking the same value for the hyperparameters. Once we replicate it, we can get the exact members and field stars using their model. Then we can compute the new metrics and compare them with the metric of our analysis. Ideally every peer-reviewed paper provides enough details in its methodology section so that one can replicate the result. Because they also provide the member group data in archive, we can test the robustness and accuracy of our replication using this member group. Thus, this hypothesis is also testable in practice.

3.2. Workflow

Figure 3.1 shows a clearer picture of how I am planning to use both supervised and unsupervised model together. Firstly, I will take the data from the GAIA early data release 3 (EDR3) for a smaller region around the cluster center. Then using an unsupervised model, I will get the labels (i.e., membership probability or binary output) for each of the stars. Then using the same features (i.e., input variables) and labels from that smaller region, I will train my supervised model. Lastly, the trained supervised model will be used with the features of the stars from the larger region around the cluster center to get the predictions for their membership probability.

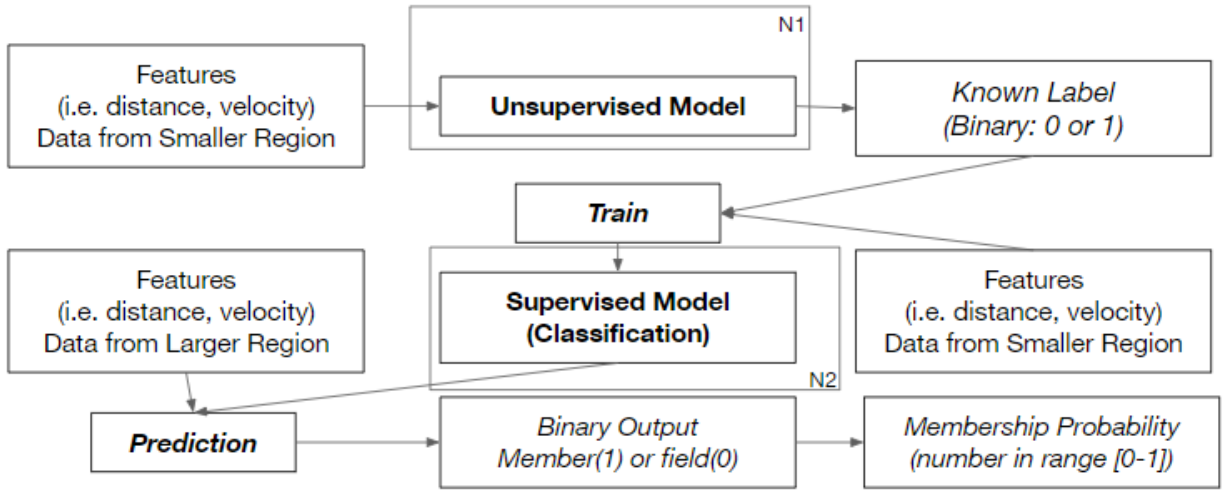


Figure 3.1: The flowchart of my project shows how I am planning to use both unsupervised and supervised methods (Pathway - 2). Several (N1) unsupervised models will be applied to a smaller region to get a binary output and the most optimal model will be chosen. Then several supervised classification models will be trained using the binary output as our training data, compared using the chosen metric to select the most optimal one. Finally, we will convert the binary output to membership probability as discussed in the text.

As I discussed in the background section the astrometric variables are more useful to determine the cluster members or distinguish between member and field stars. All members have

similar astrometric values while the field stars are randomly distributed. So, I will mostly focus on using the astrometric variables as my input variables.

The expected final output from my combined model is the membership probability, PMemb, that is a continuous variable in the range [0,1]. Once we get this probability, we could apply a cutoff to determine the possible member and field star. We will talk more about how to choose an optimum cutoff later. The reason, why we want to have a continuous variable PMemb instead of a binary variable, is the following. In star cluster study, it is common that people use the members of a cluster determined by others to analyze further properties of the cluster. Depending on the problem or hypothesis, one might want to use a more strict or more lenient cutoff to determine members. Now if the output is only binary (member or field), they cannot apply any more constraint. But if we could provide the membership probability, it will allow them to apply a more strict or lenient cutoff according to their need. Therefore, we will provide the membership probability for each star. There are two pathways I could follow to ensure a continuous variable as my final output.

In the first pathway, I could use a supervised regression model. I need to ensure that the labels used to train my supervised model is a continuous variable in the range [0,1]. And these labels are the output of my unsupervised model. So, I need to choose only those unsupervised models, which can output continuous variables. But most of the unsupervised methods are designed for clustering, thus they only provide a binary output (i.e., member or non-member). So, we need to come up with a way to get a continuous variable from the model. One way I was thinking is that we can bootstrap our data, i.e., take several samples with replacement with a sample size equal to our data size. In each iteration, we will get a binary output for each star.

Then we can use estimate the membership probability of a star as the percentage of time that star is classified as a member:

PMemb = number of times classified as member / number of total iterations

Another possible approach was used by Cantat-Gaudin et al., (2020) while using UPMASK, an unsupervised model that outputs binary variables. Every feature (i.e., *pmra*, *pmdec*, *parallax*) of the stars have their associated error value in the GAIA EDR3 data. So instead of a single number, we can approximate the feature values from a normal distribution with mean equals to their value and standard deviation (SD) equals to their error value. Now in each iteration, the feature values will be slightly different, and this difference will be larger for more noisy data (i.e., data with larger error). Then again, we can make several hundred iterations and estimate the probability of membership similar as before:

PMemb = number of times classified as member / number of total iterations

But there was a problem with both strategies. For the first strategy of bootstrapping data, the problem is, an unsupervised model is not dependent on the initial dataset. So yes, every time we would bootstrap, we would get a slightly different dataset, but our model will always be the same. Thus, the output of the models will also be the same. If a star is identified as a member for the first time, every time it will be in the bootstrapped dataset, it will always be identified as a member unless we change any of the model parameters. So, bootstrapping will not help get a probability membership.

The second strategy will work at least to give us some quantitative numbers. If the uncertainty of a single data point is higher, its value will be different in each iteration. For a precise data point, this deviation will be almost negligible. But a bigger problem is that if a data

has a large uncertainty, why would we use the data point in the first place? It is preferable to remove the noisy data during the preprocessing. Once we remove the noisy data, the deviation for all the data points will be almost negligible in each iteration. Thus, this strategy will also not work perfectly.

Therefore, in this paper I want to use the second pathway to get PMemb as a final output. In this pathway, as shown in Fig 3.1, we would use an unsupervised clustering algorithm which will use the input variables to give a binary output (i.e., member or field). Now as our label is binary, we can only use the supervised classification models. We will first train this supervised model using the binary label and the input variables (i.e., features) of the initial smaller dataset. Then we will apply this trained model in the larger dataset to predict their binary output. Now many of the supervised classification models (including Random Forest and SVM) also provide a probability value for each data point to be into a specific class. If our chosen supervised model gives this output, we can define PMemb as the probability that a star is part of the member class. Now if the supervised classification model does not provide any such output, then we can follow the first strategy: bootstrapping the dataset. This time the difference is, all supervised models are trained using the training data, thus it is dependent on the training data. Every time, we will bootstrap, our training data will be slightly different, thus the trained model will also be slightly different. As a result, the binary prediction of our target stars (i.e., stars from the larger region) will also differ slightly. If we do bootstrap for a large enough iteration, we can estimate the probability of a star to be a member as,

PMemb = number of times classified as member / number of total iterations

3.3. Evaluation Metric

One important point in this project is to define a specific metric to evaluate the performance of the models in distinguishing between member and field stars. This evaluation metric can also be used to hyper tuning the model parameters which we will discuss later. As explained in the previous subsection, we will use an unsupervised clustering model and a supervised classification model. Thus, we will focus on finding the metrics that work better for classification.

For a supervised model, defining a metric is easier as we would already have some training data with output labels. So first we can divide our training data into *train subset* and *test subset*. Then we will use only the *train subset* to train our model. Once our model is ready, we will use the features (i.e., input variables) of the *test subset* to get predictions for the *test subset*. Then we can compare this prediction with the actual labels of the *test subset*. The most common metrics used in classifications are accuracy, recall, precision.

Accuracy is the naivest approach as it determines how much percentage of the data is classified correctly as either positive or negative. Recall helps to minimize the false negatives. Recall is defined as (true positive)/ (true positive + false negative). The lower the false negative, the higher the recall score. Then precision is defined as (true positive)/ (true positive + false positive). It tries to minimize the false positive; the lower the false positive, the higher the precision score.

In open cluster studies, it is relatively okay if we predict members as field stars. This will lower the total number of members and thus may limit how much further analysis we can do using this cluster. But as normally we do not use the field stars in further studies, we will not have erroneous results. But if we classified a field star as a member of a particular cluster, then in

further analysis it would be considered as a member, and we would get misleading or erroneous results from those studies. So, it is not okay to classify field stars (negative) as a member (positive), thus we want to lower the false positive as much as possible. Therefore, I chose precision as my metric of choice while working with a supervised model.

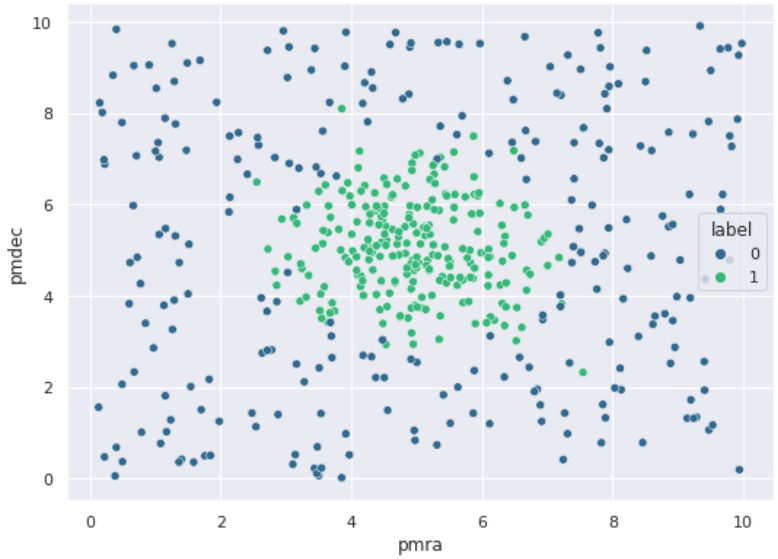


Figure 3.2: A simulated data showing how the member and field stars should ideally be distributed in their variable space. Here label = 1 are the members, which are compact and normally distributed, while the field stars are distributed all over places like a uniform and random distribution.

For an unsupervised model, it is trickier to come up with quantitative metrics, especially as we do not have a ground truth (i.e., test labels) to compare with. One common metric used for clustering, when we do not have ground truth, is the silhouette coefficient. Suppose a is the mean distance between a sample and all other points in the same class and b is the mean distance between the sample and all other points in the next class, then silhouette coefficient for the sample point is defined as, $s = \frac{b-a}{\max(a,b)}$. So, it will be between -1 to 1, where higher value means better clustering. The mean of all the silhouette coefficients is the silhouette score of that dataset. Now the problem is that the silhouette score assumes that all clusters are dense and well separated, which is not true for our case. We have a member set, which should be compact and

dense in all the feature variable space. But our field stars are random and uniform in all feature space and ideally surround the member set from all sides as shown in Figure 3.2. So, suppose we have a pretty good separation of member and field stars like in Fig 3.2, still for a member star the value of b would be close to the value of a, as the field stars are all around the member set. So still the silhouette score will be close to 0.

Ideally because the members are compact, their standard deviation (SD) should be quite small. On the other hand, the field stars are uniformly distributed, so their SD should be very large. Thus, SD could be more useful than comparing distance among clusters. So, I propose the following metric by modifying the silhouette score to evaluate performance for clustering by an unsupervised model.

Suppose there is a total of *K* features and the SD of the feature *i* for field and member group is denoted by $SD_{i,field}$ and $SD_{i,member}$. Then, the Modified Silhouette Score (MSS) is defined by,

$$MSS = \frac{1}{K}\sum_{i=1}^{K}\frac{\left(SD_{i,field} - SD_{i,member}\right)}{max(SD_{i,field}, SD_{i,member})}$$

For a well performed model, we would expect to see the members will be distributed normally with a very small standard deviation and the field stars will be uniform, thus having a high standard deviation). In this case, we would have $SD_{field} \gg SD_{member}$, therefore the numerator will be $SD_{field} - SD_{member} \approx SD_{field}$. This will result in a MSS value very close to 1. On the other hand, for a poor performed model, the member and field stars both will have similar random distribution. Thus, the numerator will be close to 0, resulting in a MSS value around 0. One special case is when the predicted field star group shows a stronger normal distribution (thus having low SD), but the predicted member group is distributed randomly (a

larger SD). In this case, the numerator will be $SD_{field} - SD_{member} \approx -SD_{member}$ and the MSS value will be around -1. So, a strong negative MSS value will likely indicate that the model was able to distinguish well between member and field stars, but it mislabeled the groups. The predicted member group is the field star group and vice-versa.

Now, I also want to empirically check the effectiveness of MSS metric for unsupervised models. I created multiple simulated datasets and apply an unsupervised model (Gaussian Mixture Model) to predict members and field members. Then I calculated the MSS metric for each of the simulation and sorted the simulations according to their absolute MSS value. After that, I visually inspected if the performance of the unsupervised model to distinguish the group looks better for the simulation with higher MSS value. This animation shows in a sorted order how the MSS metric changes for different simulation performance. From the visual inspection, it is evident that for good-performed simulation, the MSS metric is usually around or more than 0.80. Fig 3.3 shows a sample of simulated model performance for low, medium and high MSS value. We can see that MSS worked pretty well to distinguish the poor, moderate and good-performed models. Poor models, where the predicted members and field stars are not very distinguished, have low MSS value, around lower than 0.30. The moderate performed model, where we can see a pattern of normal members and uniform field stars though the clustering is not perfect, have a medium MSS value. Lastly, good-performed model, which clustered both group with more efficiently, have higher MSS value.

Another useful metric would be to compare our findings with the past literature. The most recent all-sky catalog of predicted reliable members of all open clusters using the GAIA DR2 (data release 2) is the one by Cantat-Gaudin et al. (2020). I will use it as my benchmark. As GAIA EDR3 provides more precise astrometric data, it is possible that we can find a member

which is earlier classified as a field star due to lack of precise measurement. So, while comparing, we will mainly focus to see how many members identified by the Cantat papers are also identified as members in our analysis.

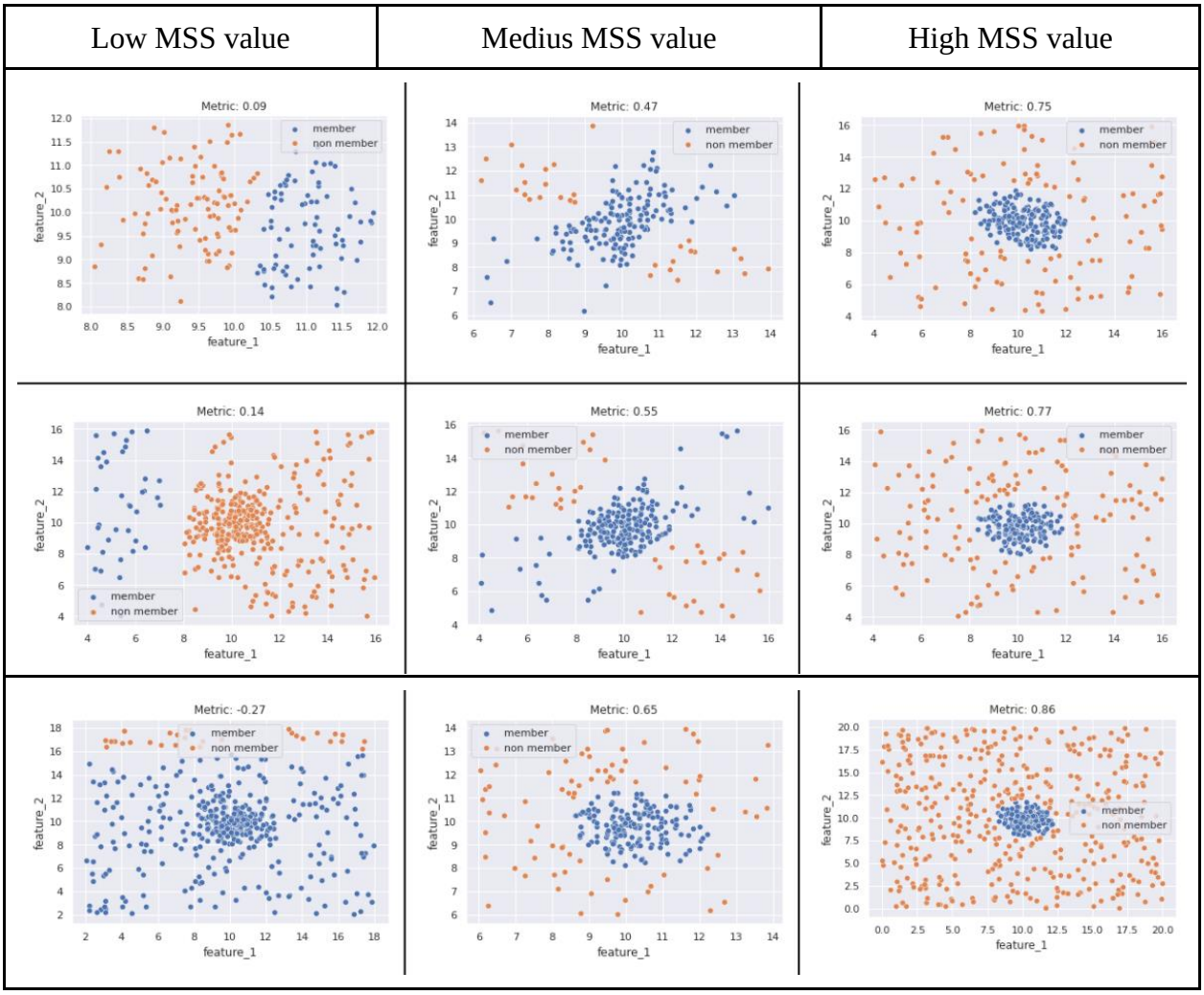


Fig 3.3: MSS values for a range of different model performance. We can see that the poor, moderate and high performed models usually have low, medium, and high value of MSS. This suggests that MSS could be a good metric to evaluate the performance of unsupervised model.

Lastly, one visual metric that has been used very widely for open cluster membership is to compare the color-magnitude diagram (CMD). Ideally the CMD for members should follow the theoretical HR diagram, where the CMD for field stars should be random. One thing to note is that while comparing CMD, we are interested mainly in the shape of the diagram. In a

theoretical CMD, we usually have absolute magnitude (related to absolute energy radiated per second), whereas in observed CMD we have apparent magnitude (how bright it looks from us, which depends on the distance of the star). So, there is usually a shift in the y-axis. Also, the color in the x-axis can be defined using different filter bands, so the exact values may change. But the overall shape should resemble the theoretical CMD.

4. Gaussian Mixture Model (GMM)

4.1. GMM Introduction

Gaussian Mixture Model (GMM) is a parametric ML model. The primary assumption behind GMM is that the observed dataset is generated from a mixture of two or more Gaussian (i.e., normal) distributions with unknown distribution parameters. The most important input parameter of GMM is the number of components, k (i.e., number of gaussian distributions that generated the data). If we have a dataset of N data points, $\boldsymbol{X} = \{\boldsymbol{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}\}$ and each data point has M different features (i.e., our feature space is M-dimensional), then the probability distribution of the data points x is defined as,

$$P(\boldsymbol{X}) = \sum_{i=1}^{k} w_i \, G\,(\boldsymbol{X} \mid \boldsymbol{\mu_i}, \boldsymbol{\Sigma_i})$$

Where, w_i is the weight of i'th Gaussian distribution, and it satisfies the following equation:

$$\sum_{i=1}^{k} w_i = 1$$

$G\,(\boldsymbol{X} \mid \boldsymbol{\mu_i}, \boldsymbol{\Sigma_i})$ is the i'th Gaussian component, and $\boldsymbol{\mu_i}$ and $\boldsymbol{\Sigma_i}$ are the mean vector and the covariance matrix of the i'th Gaussian component. As each data point is M-dimensional (i.e., have M features), $\boldsymbol{\mu_i}$ is a vector of size M and $\boldsymbol{\Sigma_i}$ is an $(M \times M)$ matrix. For one-dimensional data points, the probability distribution of a Gaussian distribution is defined as,

$$G(x|\mu, \sigma^2) = \frac{1}{(2\pi)^{\frac{1}{2}}\sqrt{\sigma^2}} e^{-\frac{1}{2}(x-\mu)^2 \cdot (\sigma^2)^{-1}}$$

where, μ and σ^2 are the mean and variance of the dataset.

So similarly for M-dimensional data using vector and matrix properties, we can write,

$$G(\boldsymbol{X}|\boldsymbol{\mu_i}, \boldsymbol{\Sigma_i}) = \frac{1}{(2\pi)^{\frac{M}{2}}\sqrt{|\boldsymbol{\Sigma_i}|}} e^{-\frac{1}{2}(\boldsymbol{X}-\boldsymbol{\mu_i})\cdot\boldsymbol{\Sigma_i}^{-1}\cdot(\boldsymbol{X}-\boldsymbol{\mu_i})^T}$$

From our dataset, we have the values for $\boldsymbol{X} = \{\boldsymbol{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}\}$, where each data is M-dimensional. In a GMM model, we need to determine the unknown distribution parameters ($w_i, \boldsymbol{\mu_i}$ and $\boldsymbol{\Sigma_i}$) for each of the k Gaussian distributions.

Once we have those components, we can denote the probability that an M-dimensional data point $\boldsymbol{x_i}$ belongs to the component C_k as,

$$\gamma_{ik} = p(\text{component} = C_k \text{ given data} = \boldsymbol{x_i})$$

Now, using Bayes' theorem:

$$\begin{aligned}
\gamma_{ik} &= p(C_k|\boldsymbol{x_i}) \\
&= \frac{p(\boldsymbol{x_i}, C_k)}{p(\boldsymbol{x_i})} \\
&= \frac{p(C_k)\cdot p(\boldsymbol{x_i}|C_k)}{\sum_{j=1}^{K} p(C_j)\cdot p(\boldsymbol{x_i}|C_j)} \\
&= \frac{w_k \cdot G\ (\boldsymbol{x_i}|\boldsymbol{\mu_k}, \boldsymbol{\Sigma_k})}{\sum_{j=1}^{K} w_j\ G\ (\boldsymbol{x_i}|\boldsymbol{\mu_j}, \boldsymbol{\Sigma_j})}
\end{aligned}$$

Here in the second line, we used the Bayes theorem to express the conditional probability in terms of joint (in the numerator) and marginal (in the denominator) probabilities. Then in the 3rd line, we expressed the marginal probability of the denominator as a summation of conditional probabilities. The marginal probability of being in the component C_k is equal to its mixture weight, w_k. Lastly, as we assumed that the underlying distributions are normal distributions, we can represent the conditional probabilities by normal (i.e., Gaussian) probability distributions with corresponding distribution parameters. Note that, the denominator of the last

line is simply the probability distribution of the GMM model $P(\boldsymbol{X})$ that we defined as the first equation.

In a GMM model, these unknown parameters ($w_i, \boldsymbol{\mu_i}$ and $\mathbf{\Sigma_i}$) are determined by the expectation-maximization iteration algorithm (Lee et al., 2012). First, we randomly (or by making an educated guess) estimate the unknown parameters. Then in the expectation step (E-step), we determine the GMM probability distribution $P(\boldsymbol{X})$. Then for all data point $\boldsymbol{x_i}$, we calculate the probability of them belonging to (i.e., generated from) each of the cluster C_k. In other words, we determine the values of γ_{ik} for all values of i and k. Next, in the maximization step (M-step), we recalculate the unknown parameters ($w_i, \boldsymbol{\mu_i}$ and $\mathbf{\Sigma_i}$) using the γ_{ik} values from E-step. Once we get updated parameter values, we go over E-step again. We iterate this loop of E-step and M-step until the algorithm converges or we reach a maximum number of iterations. An EM model is guaranteed to converge but it may converge to a local optimum instead of a global optimum (Balakrishnan et al., 2017). So, it is recommended to run GMM several times with different initial conditions and choose the best one.

In sci-kit-learn, the most important input parameter for the GMM model is the number of components, k. We can also input the number of initializations to perform. As an output, the model will return the probability of each point to be in each of the k gaussian components i.e., γ_{ik} for all values of i (data points) and k (number of components).

There are two ways we can optimize the input parameter k. If we have good prior knowledge of the dataset or the context of the problem, this can inform us about how many different components or groups are present in the dataset. If we cannot say it from prior knowledge, then we can optimize the value of k using metric evaluation. We can run GMM with different values of k and calculate our chosen metric (the most common one is the silhouette

score) for each of the models. Then we can choose the best model i.e., the model with the highest metric value.

For our problem, we want our primary dataset of stars to divide into two different groups: cluster member stars and field stars. So, we will use a 2-component GMM i.e., the number of components k = 2. Before diving in, we also need to consider how strongly our specific problem satisfies the other assumptions of GMM.

4.2. GMM in Membership Determination

One of the primary assumptions of GMM is that the given dataset consists of a mixture of two or more normal distributions. In our case, from the context, we would have two different groups of stars: member stars and field stars (i.e., non-member stars). The features i.e., the input variables (*pmra*, *pmdec*, *parallax* etc.) for all member stars should stay very close to each other, thus we can say that members are normally distributed (i.e., Gaussian) in the feature space. But the field stars should be random and uniform and should not follow any normal distribution.

So, our dataset, especially the field star group, does not satisfy the GMM assumption that all groups are normally distributed. Gao (2018a) and Agarwal et. al (2020) mentioned this problem and showed that GMM will not work especially when we have a large sample area compared to the original cluster size. In that case, we would have many field stars. As the number of these non-normally distributed field stars increase in the dataset, GMM will start to provide bad results.

Cabrera-Cano & Alfaro (1990) and de Graeve (1979) pointed out that, one way we can still use the models like GMM, which assumes two normally distributed groups, is by ensuring the following condition:

1. The ratio of the number of members and number of field stars should be higher (i.e., a low number of field stars in the sample area). This will ensure that the member group is the primary group of stars in the dataset and the other group will consist of a smaller number of stars. Then the GMM model will find a prominent normal distribution for the member stars.
2. In our feature space, the peak location of the field star distributions and member distribution should be different. Otherwise, the GMM model cannot distinguish between these two groups of stars.

Now, one way we can ensure the 1st condition is to apply GMM in a smaller region and with a smaller range of all feature variables. If our range is too wide, then we would have many field stars, and the member group will not be the primary distribution. On other hand, if our range is too small, then we would not have many field stars and GMM will try to find two separate groups within the members.

For the 2nd condition, it is unlikely that the peak of member and field star distribution will be the same in the feature space. But if it happens, then we can try to lower our sample area to reduce some of the field stars. But as we just discussed, there is a limit on how small an area (or feature space) we can choose while still maintaining a good output of GMM.

One important point to note here is that one common way to find a dataset of stars from the archival data is to use a cone search. We defined a specific position in the sky (i.e., *ra* and *dec)* and a search radius, r. Then we can extract all the stars that lie around that position within the given search radius. In a cone search, we always have a large density of stars at the center and the number density decreases as we go away from the center. Thus, for a small search radius, the field stars, and the member stars both have the highest number density in the center, i.e., the

peak of their distribution in *ra* and *dec* overlaps and breaks the 2nd condition. As we continue to increase the search radius, the distribution of the field stars starts to become flatter and uniform and the distribution for member stars becomes more gaussian. But we cannot use a large search radius for a GMM like model, as that will increase the number of field stars and break the 1st condition. So, it is better not to use *ra* and *dec* as one of our input features in GMM. Instead, we can use a smaller search radius to ensure that all the stars in the dataset are close to the cluster center.

4.3. Testing Cutoff with Simulated Data

Now we will try to understand the 1st condition a bit better. Earlier I mentioned that one way to ensure that we have a smaller number of field stars is by taking an optimum range in each of our features (i.e., independent variables). We used a simulated dataset to check how changing the range of the features influences the performance of the GMM.

For better visualization, I generated the dataset using only two simulated features: *feature 1* and *feature 2,* as an analogous to *pmra* and *pmdec*. First, I made a square grid of a constant size (size = 20). The member dataset is generated using a normal distribution with its center located at the center of the grid and with a constant standard deviation (std = 1) in both axes. The field star is generated using a uniform distribution over the entire grid. Initially the number of members and field stars are 200 and 300 respectively. In our simulated dataset, we have the *feature 1* and *feature 2* values for all stars and their true labels as 0 (field) or 1 (member). As the field stars are generated using uniform distribution, some of them also overlap with the central normally distributed member region. So, any data point that is closer to *2std* from the center is again labelled as 1 (member).

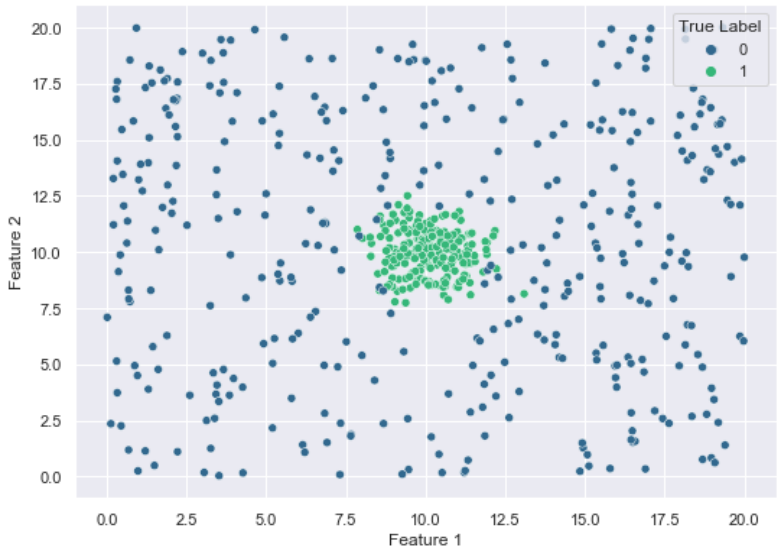


Figure 4.1: A sample simulated dataset. Features 1 and 2 are just randomly generated features as an analogous of *pmra* and *pmdec*. For member group the features are generated from a normal distribution whereas for field stars they are generated from a uniform distribution.

Next, we want to introduce the term "*half-width*". Ideally if the half-width for a given variable is *hw,* and the chosen center of the variable is x, then we will only take $x \pm hw$ values: the chosen range for the variable is *[x-hw, x+hw]*. In our simulated data, for simplicity we took the same std for both variables, which will not be the case for real data. So, we need to apply the half-width cutoff separately for each feature (or the more important ones).

First, we set the number of field stars as constant. Then we change the value of *hw* from 2std to 10std. For 2std, we would mostly have the member stars, so GMM may not work properly and for *hw* around 10std, there would be too many field stars, thus GMM may again provide bad results. Someplace in between GMM should work better. As a measurement of the GMM performance, we used the modified silhouette score, MSS.

In Figure 4.2, we can see the results for 3 different values of half-width. In the table the 3rd column shows the true labels of the star, 1st column shows the two groups generated by GMM and the 2nd column only shows the member and non-members (currently defined as PMemb 0.6 and 0.4 respectively).

Group Label by GMM	Member (PMemb >= 0.6) Non-member (PMemb <= 0.4) (Using PMemb from GMM)	True Label

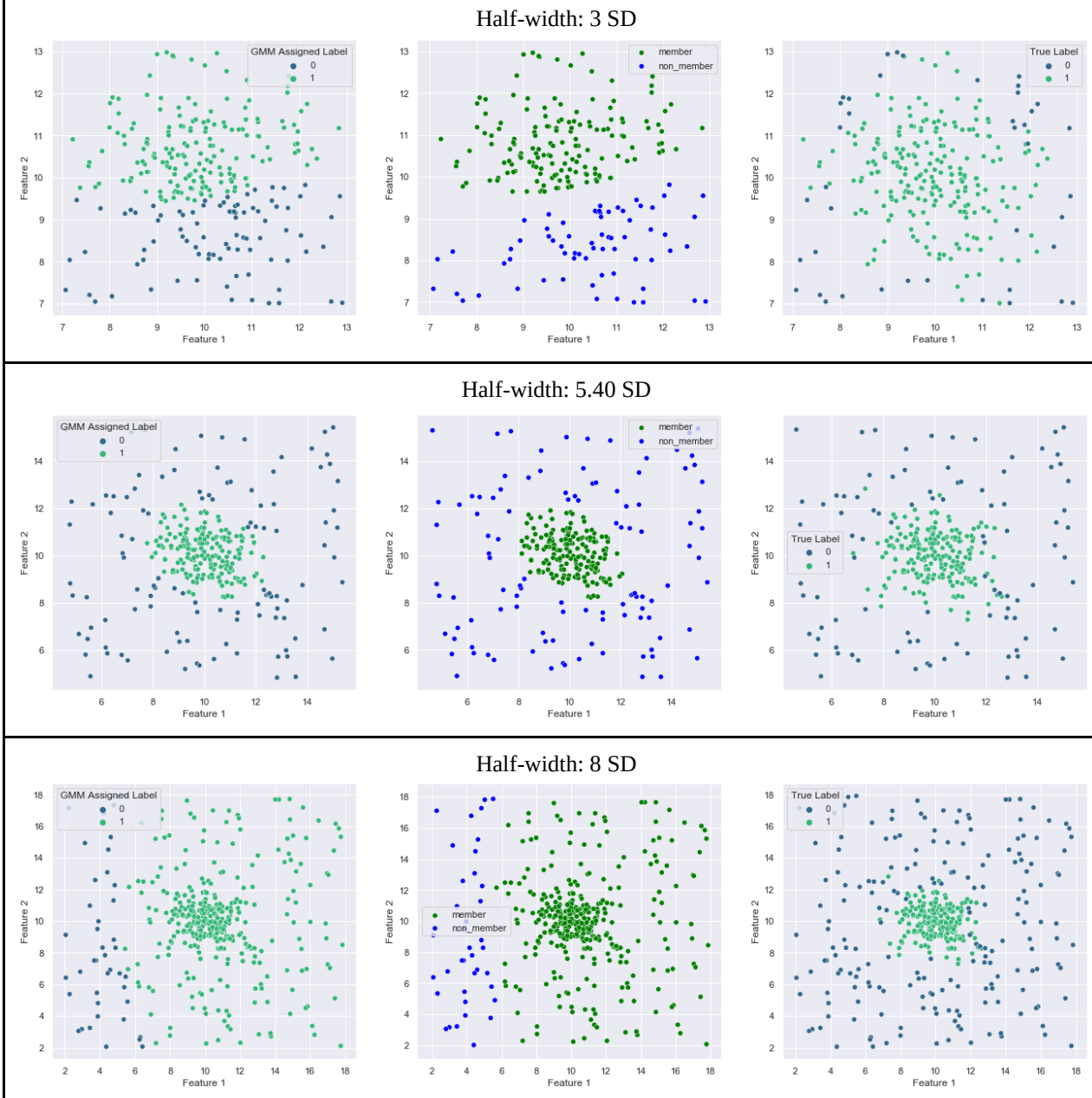


Figure 4.2: Performance of GMM for a sample of simulated data with different value of half-width. We can see that for smaller half-width, GMM fails to find the central normal distribution and divide the data almost into halves. Again, for very large half-width, due to the presence of more field stars, GMM performed very poorly. But for an optimal half-width value, GMM was able to distinguish the member and field star efficiently.

We can see that for a smaller half-width, there are only a very few field stars and GMM tries to separate the member stars into two different groups. For a larger cutoff, GMM fails to capture the prominent distribution. Only for an optimal range of *hw* values, GMM works perfectly.

Now to generalize our result a little more, we run the system for 40 different values of hw from 2std to 10std. Each of the systems is run 20 times (n_{trial} = 20) and we take the average value of the MSS metric. Then we changed the number of field stars (n_{field}) while keeping the grid size and the number of members constant. This will help us to see the effect of the field star density on GMM. For each value of n_{field}, we run the system again for 40 different values of hw and with 20 iterations each time.

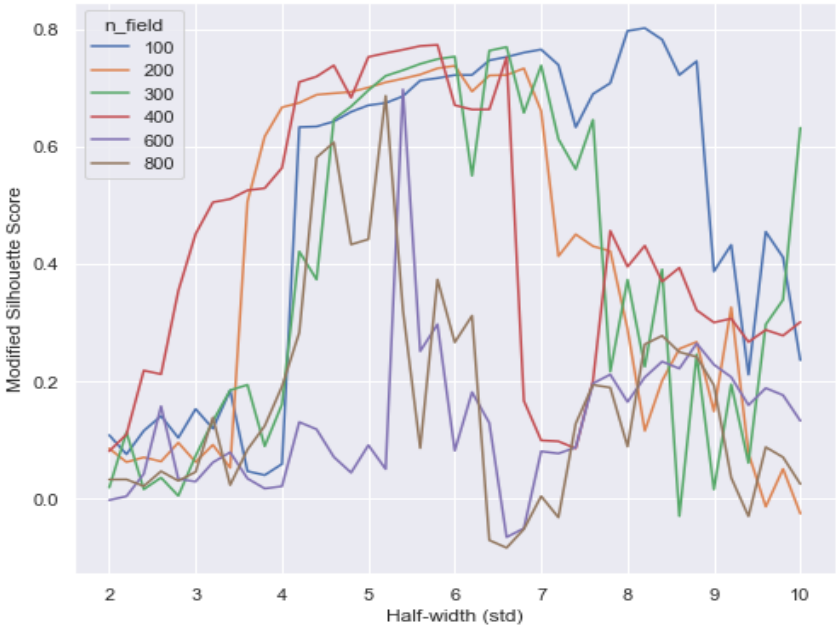


Figure 4.3: Performance of GMM, as measured by MSS and smoothed out by taking several iterations, for varying half-widths and for varying number of field stars. We can again see a common trend that the MSS value starts at very low for smaller half-width, then it increases in the middle before falling again for higher half-width value. Also, as the field star contamination increases, the optimal region of half-width value becomes narrower.

Figure 4.3 shows the result for all the analysis. We can see a common trend for all values of n_{field}. Each of them has a low MSS value at first. Then in the middle as *hw* increases to around 4std to 6std, they have a flat peak at MSS. Then again for larger hw they fall back for a lower value. So overall qualitatively we can say that for GMM to work properly, we always need to find an optimal cutoff, especially if the variable is not normally distributed for both members and field stars.

Another point to note from Fig 4.3 is the influence of the field star density. As the grid size is constant, the field star density increases with increasing values of n_{field}. We can see that for a smaller field star density, we start to get a good performance for a relatively high value of hw (5.5 std) and it stays good up to a relatively high hw value (7.5 std). It is because due to smaller number density, we need to increase the range of the variable to get enough field star in our dataset. As the n_{field} increases, the optimum region comes towards the lower values of *hw* (around 3.3 - 5.5 std). Then once the field star increases even more ($n_{field} = 800$), we never have any good performance. The best value of the MSS metric is still quite low for those very high field star density.

One important note is that we cannot come up to a robust quantitative idea on the optimum range of *hw* values just from this simulated data example. Here we considered a very simple data, with similar distribution in both axes. We need to run this kind of test for different possible case scenarios with varying std values and varying grid size and make a thorough analysis of the system for a robust quantitative result. But our goal here was to qualitatively understand whether GMM can work properly if we can find some optimum range in the feature spaces. Moreover, in real data, we would not know the field star density or the standard deviation of our member stars. Thus, we need to come up with some strategy to empirically find the

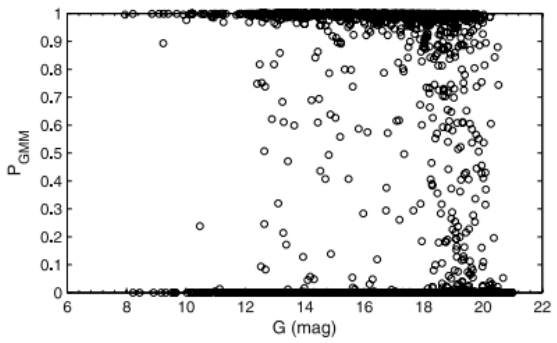
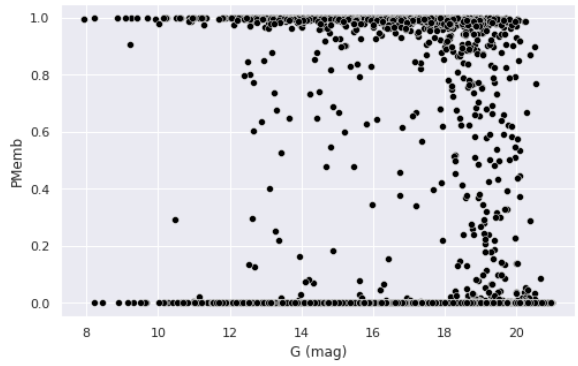
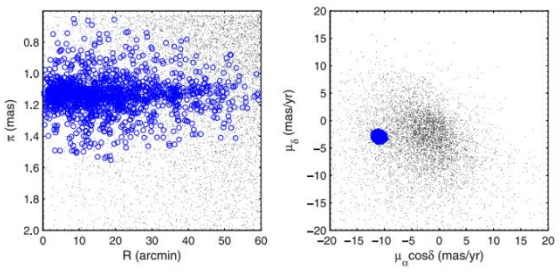
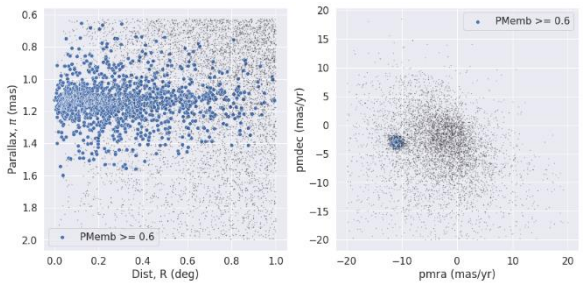
optimal cutoff. We would see one such example in the next subsection. We also found that GMM will not work if the field star density is too high. So, we need to keep our search radius small to ensure small number of field stars.

4.4. Past Paper Replication

Now, we can go over some past papers to check and understand how they considered these nuances while applying GMM in cluster membership problems. I chose the Gao (2018a) paper to replicate for two reasons. One it is one of the very few papers, where the unsupervised (GMM) and the supervised method (Random Forest) are applied together. So, I can go over the same paper again to understand the nuances of applying the supervised methods and how to combine them well. Secondly, he used the cluster M67, which is a very common and well-studied cluster, so, when necessary, we can compare with the results from many other papers.

Firstly, I will try to follow the same processes and same assumptions from the paper to ensure that I can replicate the results properly using my code. Then I will modify any assumptions or process that I think will make the analysis even better.

M67 is a close-by old cluster with a distance of 860 pc. The Gao paper first took a 2.5-degree search radius with a general filter on *pmra* and *pmdec* that they should lie between -20 to 20 mas/yr. The GMM failed to segregate the members initially, so he had to change the search radius to 1 degree and apply a special cutoff in distance: the distance of the stars in the dataset should be between 500-1600 pc. After this cutoff, he applied the GMM in the total 7312 sample stars using all five astrometric variables. I followed the same procedure to reproduce the numbers and the figures of the paper. Table 4.1. summarizes the comparison between the Gao paper and my direct replication.

	Gao M67 Paper (2018a)	Direct Replication
Sample Stars	7312	7316
n(PMemb >= 0.6)	1401	1416
n(PMemb >= 0.95)	1256	1261
n(PMemb <=0.0001)	5720	5691
Covariance matrix for field star, f		
Covariance matrix for member star, m		
PMemb vs G-band magnitude		
Parallax vs distance from the cluster center, proper motion plot		

	pmra	pmdec	ra	dec	parallax
pmra	0.965	-0.232	-0.004	0.006	-0.042
pmdec	-0.232	1.232	0.005	-0.008	-0.079
ra	-0.004	0.005	1.154	-0.013	0.010
dec	0.006	-0.008	-0.013	1.161	0.011
parallax	-0.042	-0.079	0.010	0.011	1.204

	pmra	pmdec	ra	dec	parallax
pmra	0.963	-0.232	-0.005	0.004	-0.042
pmdec	-0.232	1.232	0.005	-0.010	-0.079
ra	-0.005	0.005	1.158	-0.013	0.008
dec	0.004	-0.010	-0.013	1.163	0.007
parallax	-0.042	-0.079	0.008	0.007	1.205

	pmra	pmdec	ra	dec	parallax
pmra	0.003	0	-0.006	0	0.005
pmdec	0	0.003	0.001	-0.003	0
ra	-0.006	0.001	0.341	-0.020	0.001
dec	0	-0.003	-0.020	0.318	0.008
parallax	0.005	0	0.001	0.008	0.141

	pmra	pmdec	ra	dec	parallax
pmra	0.003	0	-0.006	0	0.005
pmdec	0	0.003	0.001	-0.003	0
ra	-0.006	0.001	0.342	-0.020	0
dec	0	-0.003	-0.020	0.321	0.007
parallax	0.005	0	0	0.007	0.150

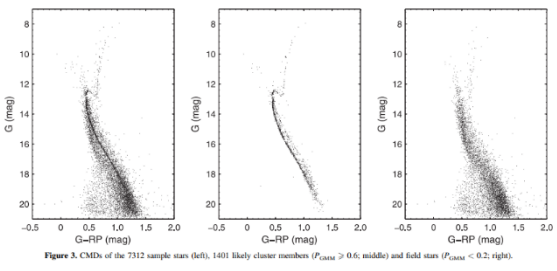
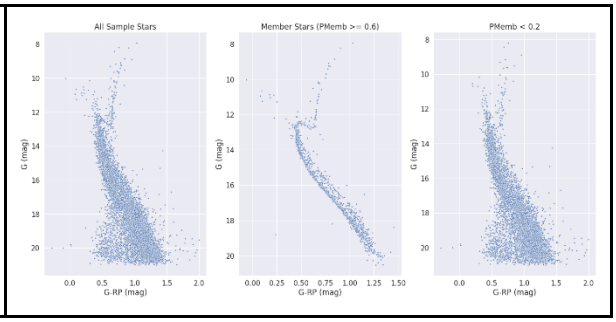
Color Magnitude Diagrams (CMD)		

Table 4.1: Comparison between Gao paper and the direct replication

We can see that the numbers and figures match quite closely. There is a slight variation in the number of sample stars. One possible reason could be that I took the coordinate of the cluster center using the *SkyCoord* library of astropy package, which uses the most recent data from SIMBAD (Wenger et al., 2000). For open cluster data, it currently uses the Cantat-Gaudin paper (2020). While Gao took the center coordinate from WEBDA (Mermilliod J.-C., 1995; Netopil et al., 2012). Thus, there could be some small variation in the center coordinate, thus changing the number of sample stars slightly.

The covariance matrix is analogous to the variance for high-dimensional data. So, we expect that the numbers in the covariance matrix should be lower for the member group and higher for the field stars. Here it satisfies the expectation as the numbers are very close to 0 for members and relatively high for field stars. Note that for *ra* and dec, even for the field stars, most of the value in the covariance are smaller, suggesting that it is close to a normal distribution.

PMemb vs G-band magnitude plot shows that GMM is quite confident about its identification of the field and member stars as most of the data have extreme PMemb values. It is always harder to observe and measure the fainter stars, thus the fainter stars usually have higher error. So, it is expected that the uncertainties should be higher in the fainter region. That is why we see that PMemb is spread all over from 0 to 1 for fainter stars.

The proper motion plot (pm plot) clearly shows that the identified member group is very different in the proper motion space compared to the field stars. There is a very important thing to note in the pm plot. Unlike the parallax space, in the pm space, the field stars are not very random and uniformly distributed. Instead, we can see that they are close to a normal distribution in the pm space. This is because most of the stars are not moving (or moving very slowly) through the sky plane. So, if we take a random set of stars and measure their proper motion in the sky, most of them will lie close to the (0,0) point and we would expect to see a Gaussian-like distribution centered around (*pmra*, *pmdec*) = (0,0). This is also the reason why we did not need to give a special half-width cutoff for the proper motions.

The CMD shows that the identified members lie very well in a color-magnitude diagram, and they resemble a theoretical CMD with an age of 4 billion year closely.

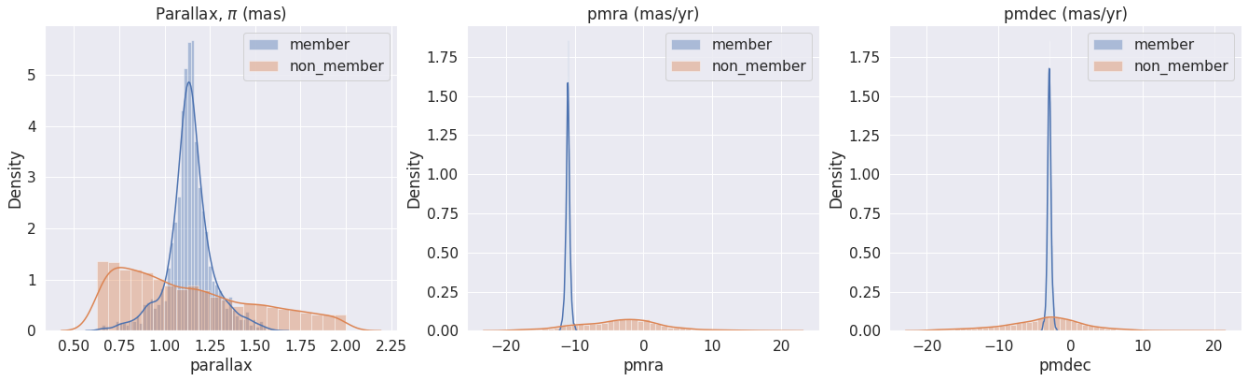


Figure 4.4: Distributions of parameters for the identified member and non-member of M67 by GMM model. In all three parameter space (*parallax*, *pmra* and *pmdec*), the member group shows a narrower gaussian distribution whereas the field stars consist of a wider range of values.

Lastly in figure 4.4, we can see that the distributions of the members are very distinctly different and form a normal distribution in all important features. The MSS metric for this direct replication result with a member cutoff of 0.6 and 0.95 are 0.698 and 0.729, respectively.

4.5. Modified Replication

Now we will add some of the modifications to the M67 membership problem to improve the analysis and compare our results again.

Firstly, as we discussed earlier, for a smaller search radius, *ra* and *dec* can sometimes hinder the performance of GMM, as both groups of stars often have a similar peak position and similar distribution. Fortunately, for a smaller radius, all the stars are very close to the cluster center, so it is not necessary to include them in our model. Thus, I will run the same analysis with only three important variables of choice (*pmra*, *pmdec* and *parallax*). Another point is that we need to get rid of noisy data as we cannot get useful information from an observation with large error. We followed the filter used by Manan et al (2020) for filtering noisy data. Photometric variables are easier to observe precisely then the astrometric variables. So, if we removed the data where the photometric error is larger, we can automatically get rid of most, if not all, of the noisy astrometric data. Thus, Manan et al (2020) took only the stars whose G-band magnitude error is smaller than 0.005.

Secondly, the cutoff used for the distance (500-1600 pc) was possibly found by a trial-and-error approach. I want to propose a more systematic way to find the cutoff using the evaluation metric, in this case, the Modified Silhouette Score (MSS). Like what we did with the simulated data, we can first find the distance of the cluster from literature, which will be the center of the distance. Then we can continue to change the half-width for the distance and every time measure the MSS metric values. In addition, we can also keep a record of how many member stars are identified. So as a first criteria, we will find the half-width where the MSS is highest. If the MSS values are very close for several half-widths (or higher than a good MSS value), then we can choose the one with the highest number of members.

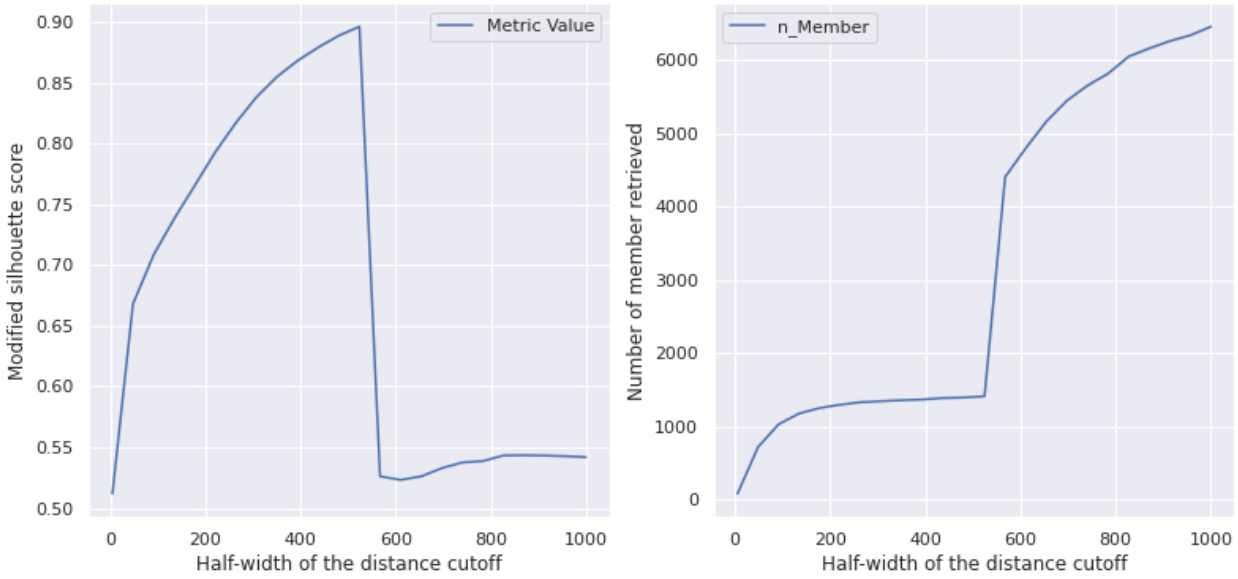
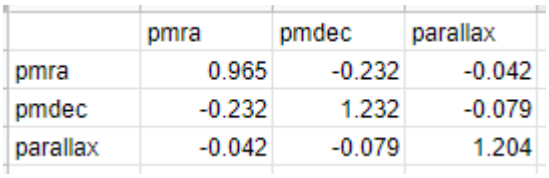
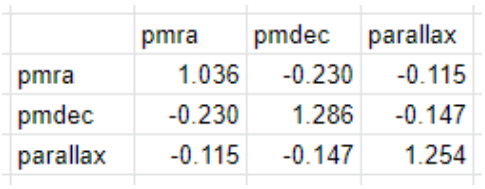
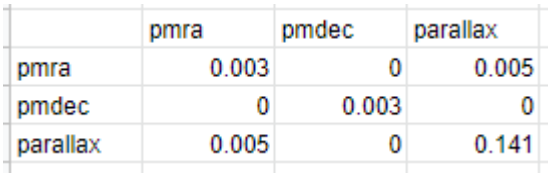
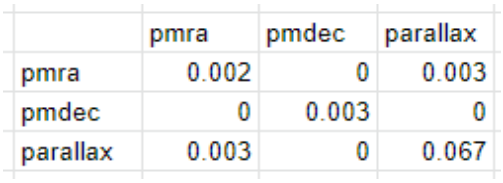
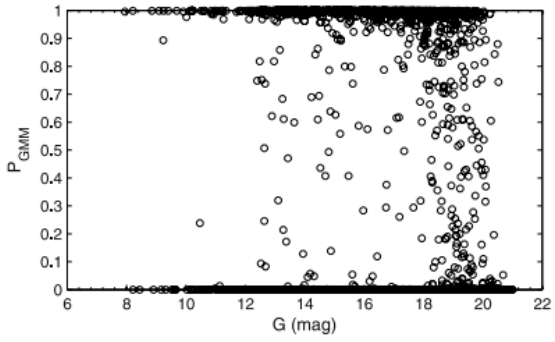
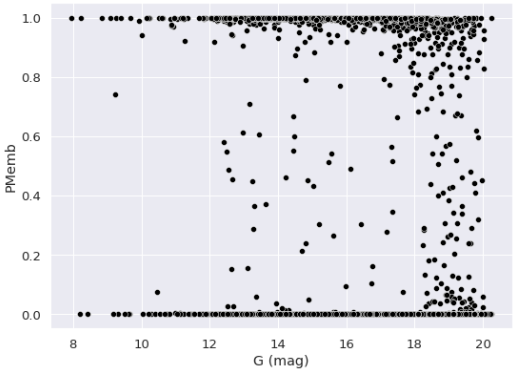


Figure 4.5: MSS value and number of retrieved members for varying half-widths of distance, measured in parsec, pc, which we can use to choose an optimum distance cutoff. We can see that as half-width increases, model performance increases up to around 550 pc, then it drops.

Fig 4.5 shows the MSS vs half-width and the $n_{members}$ vs half-width plots. We can see that the MSS keeps increasing up to a certain point, after that GMM failed to segregate the field and member stars. After that point, the number of stars in the 'claimed' member group becomes surprisingly high. This is likely a similar situation as we saw in the last row of Fig 4.1. However, using Fig 4.5, we can find the maximum MSS metric as 0.896 and the optimal halfwidth as 524 pc. So, we will keep only the stars with a distance of 890 ± 568 pc, i.e., between the range of $[302, 1478]$ pc.

Thirdly, instead of initializing only once, we will ask the GMM algorithm to run with 5 different initial conditions and choose the best one (by setting $n_{init} = 5$). This is because the convergence of the M-step of the GMM model is only guaranteed to a local optimum, not a global optimum.

The following table summarizes the findings of the modified version of the replication with the original paper. We can see that most of the numbers and figures stay very same, though our justification becomes a bit stronger in the modified version.

	Gao M67 Paper (2018a)	Modified Replication
Sample Stars	7312	6118
n(PMemb >= 0.6)	1401	1394
n(PMemb >= 0.8)	1256	1274
n(PMemb <=0.0001)	5720	4558
Covariance matrix for field star, Σ_f		
Covariance matrix for member star, Σ_m		
PMemb vs G-band magnitude		

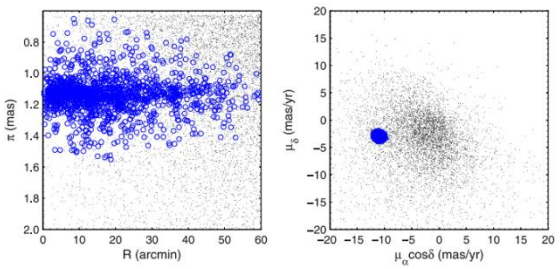
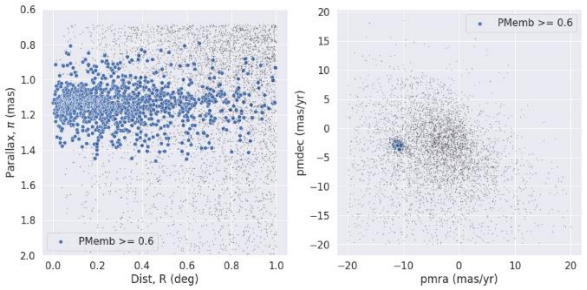
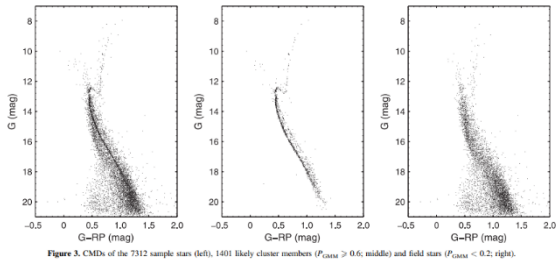
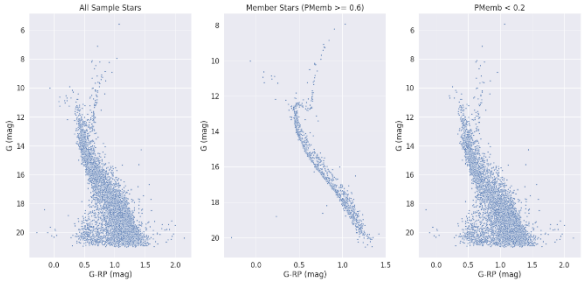
Parallax vs distance from the cluster center, proper motion plot		
Color Magnitude Diagrams (CMD)		

Table 4.2: Comparison between Gao paper and the modified replication

4.6. Threshold for Member

One of the choices that we need to make many of the times is to set a given threshold for selecting the member and nonmember. We would like to keep the final output as PMemb, so that people can use a stricter (maybe for finding the initial mass function) or lenient (maybe for finding the average distance of the cluster) cutoff according to their need. But if we need to compare the total number of members found and thus want to decide on threshold, we must justify that choice. Moreover, if we want to make a training data for supervised model, we need to choose a threshold for members and non-members. One way to justify the threshold is to check the CMD, proper motion plot and other necessary plots for different cutoff values and check visually that for which cutoff we are getting the best plots. Another proposed approach could be to use the MSS score or any other justified evaluation method. As seen in Fig 4.6, we can make a MSS vs cutoff plot and choose the cutoff based on the highest value of MSS or

maybe at which we crossed a certain threshold in MSS. The best MSS value for the modified replication is 0.91 with a 1000 half-width cutoff in distance and a 0.95 threshold in member.

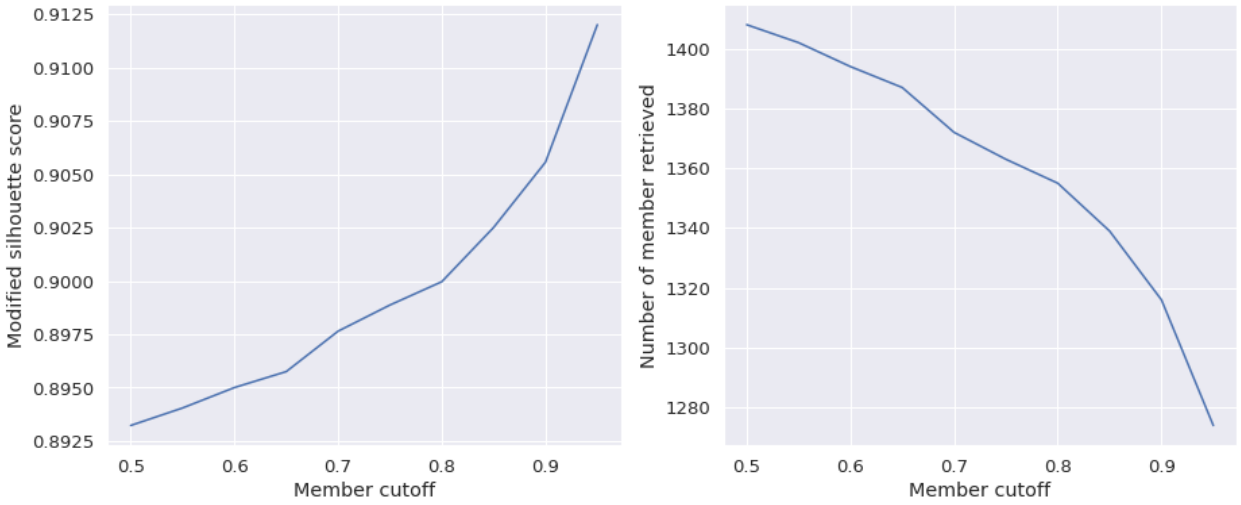


Figure 4.6: MSS value and number of retrieved members for varying member cutoff (threshold used for choosing member and non-member) for the cluster M67. We can see that the stricter the cutoff the better the model performance, though it decreases the total number of members. As we want to have few confident members, then more unreliable members, we choose the cutoff, where MSS value is largest (0.95).

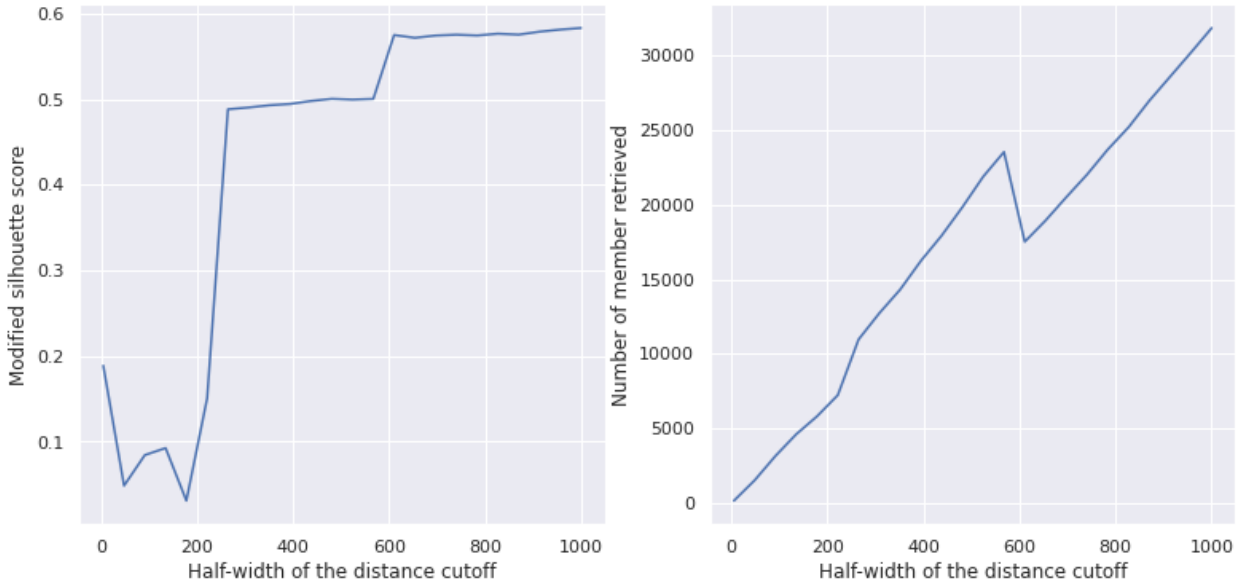
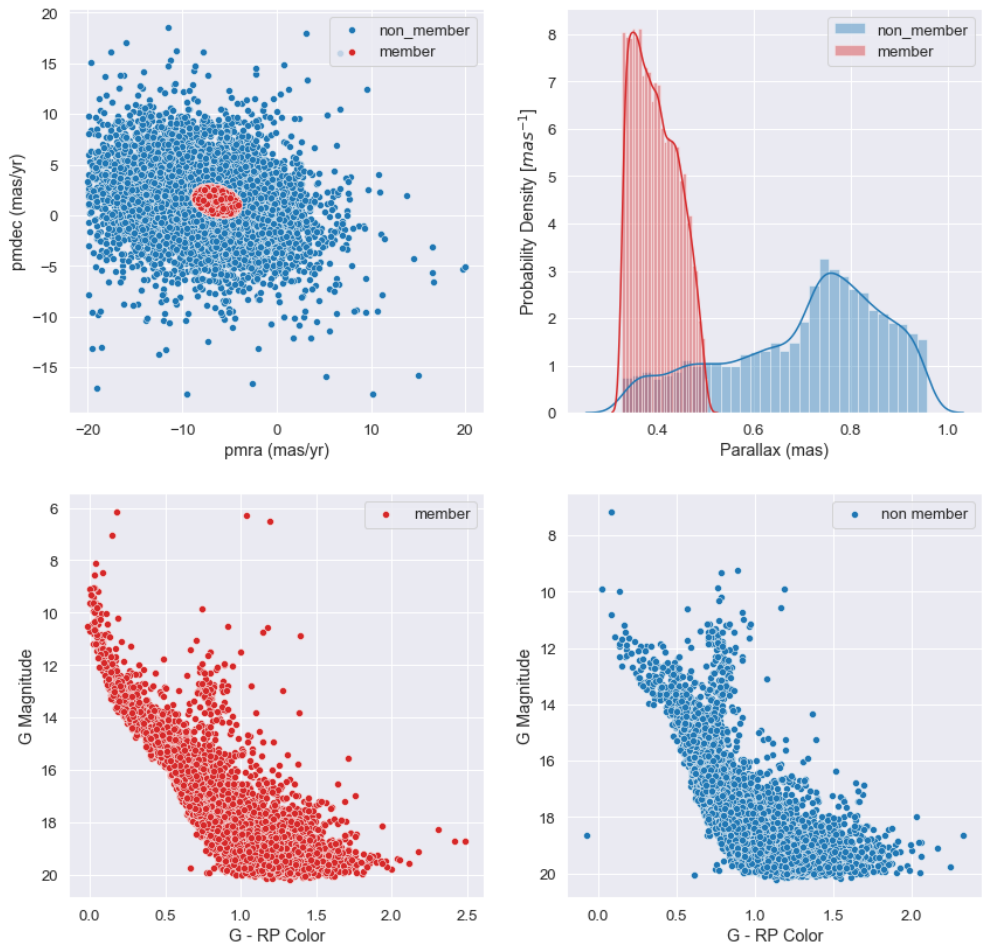


Figure 4.7: MSS value and number of retrieved members for varying half-widths of distance, measured in parsec, pc, which can be used to choose an optimum distance cutoff for the new cluster NGC 3766. We can see that MSS, the model performance metric, plateaus around 0.6. We choose the cutoff as 1000 pc as this results in more member stars without compromising the performance much.

4.7. Running GMM model in a new cluster

Now we run the GMM model for NGC 3766, which is around 2045 pc away from us. We used 0.5 deg as initial search radius and apply the same procedure as modified replication. Using the MSS vs half-width plot (Fig 4.7), we get a MSS value of 0.61 at a halfwidth of 1000. The MSS value is quite low, so the GMM model is likely to not work very perfectly. Fig 4.8 shows the distribution in different feature spaces.



MSS Metric Value: 0.78, Member cutoff 0.95, half-width: 1000

Figure 4.8: Distribution of different parameters for predicted member and non-member groups after applying the modified GMM model on NGC3766 open cluster. Compared to the field stars, the member stars more closely stays together in the proper motion space as well as in the parallax. The color magnitude diagram of the member somewhat follows the theoretical CMD, though the thickness of the CMD is larger than ideal, which suggests uncertainty about all the predicted members being the actual member.

. In the proper motion, the members lie in a very compact and almost spherical distribution, whereas the field star distribution is very broad. Again, In the parallax space, members are distributed in a narrower range around 0.4 mas, while the non-members show a very wider and flatter distribution. These two plots indicates that the predicted members are very likely to be the original member as they confirmed the assumptions of members being in a compact normal distribution.

But the CMD of the members is not very much different than the field star CMD. Though it follows the general structure of a theoretical CMD, the stars are scattered very much around the diagonal main-sequence line. This suggests that it is possible that many of the predicted members are maybe not the true members.

But this is the best performance we could get using GMM model. If we run several other unsupervised models for NGC 3766, then compare their MSS value, we may find that some other unsupervised model performs better for this specific dataset. For now, we will go over an example of a supervised model to understand how we can apply them to determine the confident members.

5. Random Forest (RF)

5.1. Random Forest Introduction

Random Forest (RF) algorithm uses an ensemble of decision tree to predict categorical (classification) or numerical (regression) target variables. Thus, it is important to first go over the characteristics of a decision tree. Decision tree model is a non-parametric model; it does not assume any underlying functions to relate the features (independent variables) with the target (dependent variable). As a supervised model, it needs a labelled training dataset to train the model. While training, it maps the relation between features and target using a set of if-then-else statements and creates a piecewise function. In training phase, a decision tree classifier splits a single parent node into two child nodes using Gini impurity. For each split, the tree needs to choose two parameters: a feature, f and a threshold t ($\theta = (f, t)$)**.** All the observations where the value of $f \leq t$, go to the left child node, and all other observations go to the right child node. Gini impurity of a node measures the following idea: if we randomly classify an observation according to the distribution of the observations in the node, then what is the probability of misclassifying that observation. Suppose we have k different classes $(1,2,3 \dots, k)$ in node m, and the number of observations for each class is denoted by $N_1, N_2, \dots, N_k$ such that $N_1 + N_2 + \cdots + N_k = N_m$. The probability of finding class i in node m is

$$p_{mi} = \frac{N_i}{N_m}$$

Then the Gini impurity for node m will be,

$$\mathrm{G_m} = \sum_{i=1}^{k} \text{probability of classifying the observation as } i \times \text{probability of the observation not equal to } i$$

$$\mathrm{G_m} = \sum_{i=1}^{k} p_{mi} \times (1 - p_{mi})$$

Here, as we classify the random observation according to the prior distribution, the classify probability is same as the class probability, p_{mi}. Gini impurity of a given split with parameter $\theta = (f, t)$, is defined by the weighted sum of the Gini impurity of both left and right child node, m_{left}, and m_{right}:

$$\mathrm{G_\theta} = \frac{N_{right}}{N_m} G_{m,right} + \frac{N_{left}}{N_m} G_{m,left}$$

Here N_{right} and N_{left} denotes the total number of observations in the right and left child nodes respectively. The lower the Gini impurity is, the lower the misclassification rate, thus the better the split. For each split, the decision tree chooses the parameter with the least Gini impurity.

There are some other important parameters that determine how long the tree will continue to split into child nodes. The root node is the original dataset with all the observations. Leaf nodes are the terminal nodes of the tree who does not have any child nodes. The height or the depth of a node is defined as the minimum connection between that node and the root node. Height or depth of a tree is defined as the maximum depth of the leaf nodes. The parameter *max_depth* limits the maximum possible depth of the tree. If the depth of a node is equal to the *max_depth,* it will not split anymore. The parameter *min_samples_split* denotes the minimum number of observations (or samples) that a node must have to split. A node cannot split if it has lesser sample. Another parameter *min_samples_leaf* denotes the minimum number of samples a leaf or terminal node must have. If a certain node has more sample than this and the depth of the node is less that the *max_depth,* then it will continue to split.

After training phase, we will have a trained decision tree where each terminal node has one or more training samples. During prediction, a trained decision tree compares the feature values of the test data with the threshold of each split and travel down from its root node up to a terminal (or leaf) node. For regression problem, the final prediction for that test data is the average value of the training samples in that lead node. For classification tree, the final prediction of that test sample is the most frequent class in that leaf node. Additionally, the prediction probability, the probability that it belongs to this class, is the fraction of samples of that class in that leaf node. For example, if that leaf node has 3 samples from class A and 2 samples from class B, the prediction will be class A with a probability of $\frac{3}{5} = 0.60 = 60\%$.

The problem with decision tree is that it become too specific of the training data, thus have a low bias but very high variance. It overfits the model and its evaluation metric are higher for the training data, but lower for an unknown test data. A small variation of the training data can result in a very different decision tree. Another problem is that the decision tree results in some piecewise function between the features and the targets, instead of a continuous or more robust relationship between them. Thus, we cannot use decision tree to extrapolate outside of the domain of training features.

Random Forest reduces the variance of decision tree model, while only a slight increase of the bias, by considering several hundred trees instead of just a single tree. It uses the bootstrapping method to increase the variation of each tree. While building each tree, it takes a random sample with replacement from the dataset with the same size of the dataset. As it is a sampling with replacement, each tree has slightly different dataset. Similarly for each tree, the features are also selected using a random sampling with replacement. How many features we want to select randomly each time is determined by the parameter *max_features.* The total

number of trees in the ensemble is selected by *n_estimators.* As each tree is slightly different than each other in terms of observations and features, the tree models are different from one another. The final prediction of the random forest is the average of the predictions from all trees of the ensemble. For classification, one way is that the final prediction is the class predicted by most of the trees. Another approach, which is used in Python's Sci-kit learn package (Pedregosa et al., 2011), is that the final prediction probability for each class will be the average of the prediction probabilities for that class by all trees. Then the final prediction will be the class with the maximum average prediction probability.

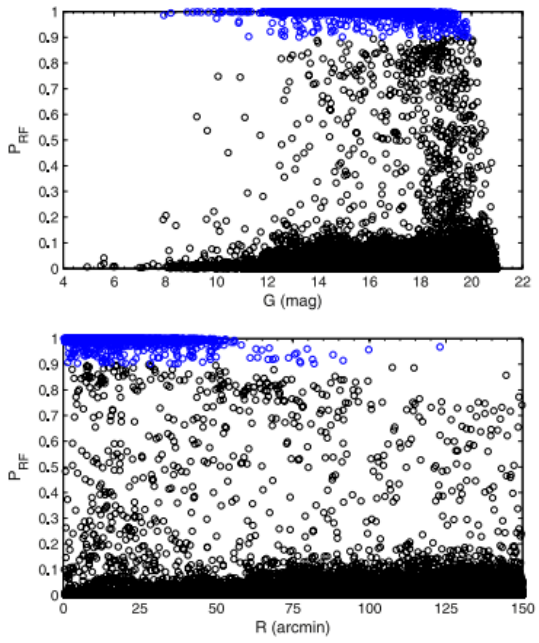
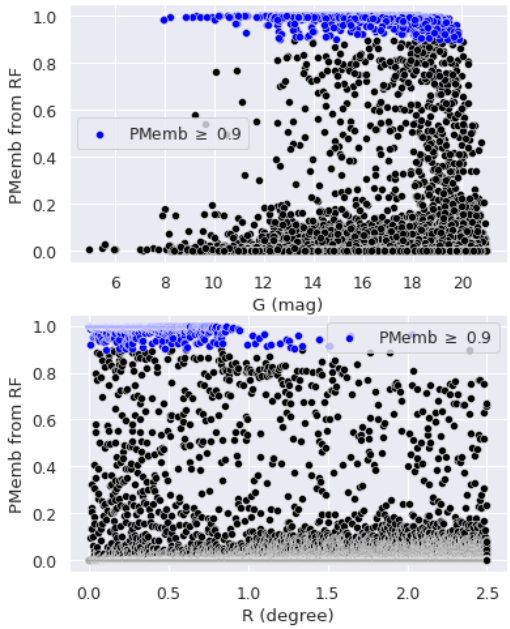
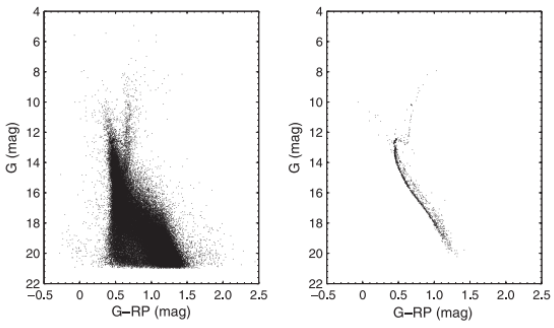
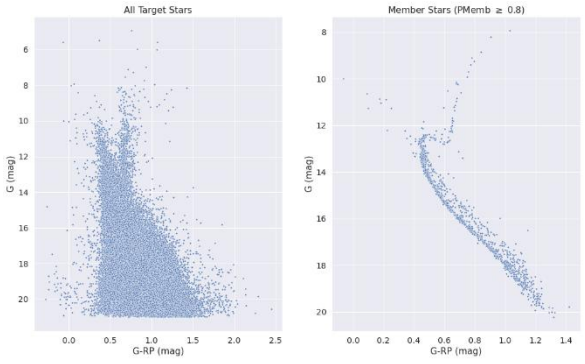
One important point to note is that the random forest still predicts using the piecewise functions generated by all those trees. Thus, random forest is still cannot work well for extrapolation. We need to ensure that the range of the new dataset features, where we want to apply Random Forest model, does not go beyond the range of the training features for that model.

As explained in the workflow (section 3.2), we want to use the stars from a smaller region around the cluster center in the unsupervised model to generate our training data. Using the training data, we will train the supervised model. Then we will apply that supervised model to a larger region of stars around the cluster center. Thus, the positions *(ra, dec)* of the target stars (larger region) are not covered by the ranges of training stars (smaller region). As random forest cannot extrapolate, we cannot use *ra, dec* as a feature for the random forest. Similarly, if we used any cutoff for the features in unsupervised model used to generate our training data (i.e., distance cutoff in GMM), we need to apply similar cutoff for the target stars while working with Random Forest.

5.2. Past Paper Replication

Once we got a good understanding of how RF model works, we can now replicate a past paper to understand the best practices while applying RF for star cluster membership. I continued to work with the same Gao paper (2018a) to replicate the use of Random Forest model using the training data generated from the GMM model. Like the replication of GMM model (unsupervised model), I would first follow the same assumptions and procedures from the paper for a direct replication. Then in next subsection, I would perform necessary modifications to improve the justifications and/or the performance of the model and compare the results. After that in section 5.4. I would apply the similar procedures for a new cluster that I chose in section 4.5. for GMM model.

While building the training data from the GMM output, he used a $PMemb_{GMM} \geq 0.95$ cutoff for selecting members and $PMemb_{GMM} \approx 0$ for non-members (or field stars). Then he used all five astrometric variables *(ra, dec, pmra, pmdec, parallax*) and six photometric variables (as explained in background: *G, BP, RP, BP-RP, BP-G, G-RP*), a total of eleven variables as features of RF model. He chose 7000 as the total number of trees (*n_estimators = 7000)*. While building each tree, the number of randomly chosen features is 4 (*max_features = 4*). All other model parameters are chosen as their default values. As a target dataset, they took all the stars around 2.5-degree radius from the cluster center. The results of my direct replication and the Gao (2018a) paper are compared in Table 5.1.

	Gao M67 Paper (2018a)	Direct Replication
Target Stars	71,117	70987
n(PMemb >= 0.6)	1502	1529
n(PMemb >= 0.8)	1361	1375
PMemb vs G Mag, PMemb vs R (distance from the cluster center)		
CMD	 **Figure 7.** CMDs of the 71,117 sample stars (left) and 1361 high-probability cluster members ($P_{RF} \geq 0.8$; right).	

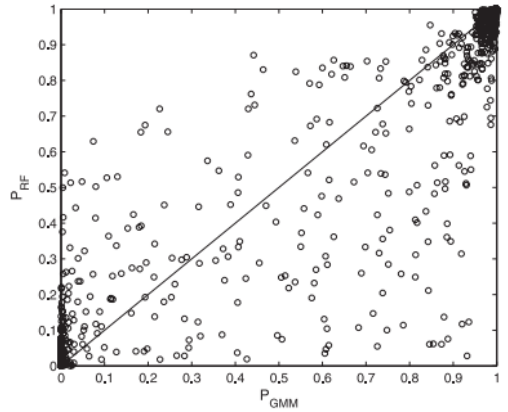
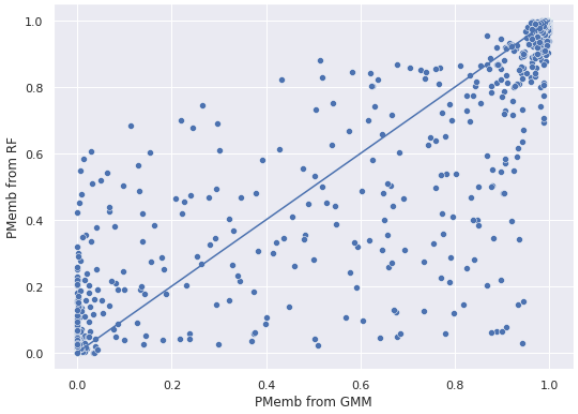
Correlation between both of the PMemb	 **Figure 8.** A comparison between the GMM and RF membership probabilities of the 7312 sample stars.	

Table 5.1: Comparison of RF results between Gao (2018a) paper and my direct replication

Here again we can see a slight difference in number of target stars, which is most likely due to the slight variation of the cluster center coordinate taken by Gao paper (WEBDA) and me (from Cantat-Gaudin, 2020). The difference is a bit larger than the GMM model (as Table 4.1), as now I took a larger search radius. The number of stars with that I found with $PMemb \geq 0.6$ and $PMemb \geq 0.95$ are also almost equal to the original paper. An additional cause for the small variation of these numbers could be the existence of some random component in RF model (for example, for sample and feature selections).

Similar to GMM model, PMemb vs G-band magnitude plot again shows that for brighter stars the model is more confident about its classification and only a small number of stars have a medium PMemb value. But for fainter stars (specially for $G \geq 18$) it becomes harder to distinguish and we have a spread of PMemb from 0 to 1. PMemb vs R plot shows that most of the confident members are inside of 1-degree regions from the cluster center. As the stars go away from the center, it becomes harder for this model to classify as member. The CMD with the predicted members follow the theoretical CMD very closely. We can clearly see the main sequence and the supergiant lines as well as the turnoff point of the cluster. The last plot shows

how the prediction probability varies for GMM and RF model. Firstly, most of the predictions is similar for both model (top-right and bottom-left corner of the plot). A number of stars, predicted strongly by GMM (high PMemb), are discarded by RF model (bottom-right), while the stars strongly predicted by RF are almost always predicted strongly by GMM. Lastly, as there were no test or held-out set, I could only measure the training precision score, which is 1.0 indicating that all members identified as the

5.3. Modified Replication

5.3.1. Feature Selection

Instead of taking all available features directly, I want to apply some feature selection techniques to choose the important topics. There are several different ways for feature selection; most common is to quantify the feature importance by some metric and choose the best ones. I explored three different types of feature importance metrics, their advantages, and limitations before choosing one for our analysis.

The default metric used in RF model and other tree-based models is 'Mean Decrease in Impurity' (MDI). As we discussed in section 5.1., at each split of the tree, the tree uses a certain feature which decreases the largest Gini impurity. To calculate MDI of a specific feature, we can consider all the splits of the tree that contains the features and take a weighted sum of the decrease in Gini impurity. The weight is determined by the fraction of samples affected by the samples. A certain split affects all the samples in the parent node. If the parent node of the split contains N_m samples and the total number of samples in training data is N_{total}, then the weight of the split is $\frac{N_m}{N_{total}}$. In the Random Forest, the overall MDI of a feature is calculated by taking the average of the feature's MDI from all the trees in the forest. Lastly, it normalized the MDI

values by dividing by the sum of all MDI to get a relative feature importance for each feature (Louppe, 2014). Fig 5.1 shows the feature importance of our RF model using MDI.

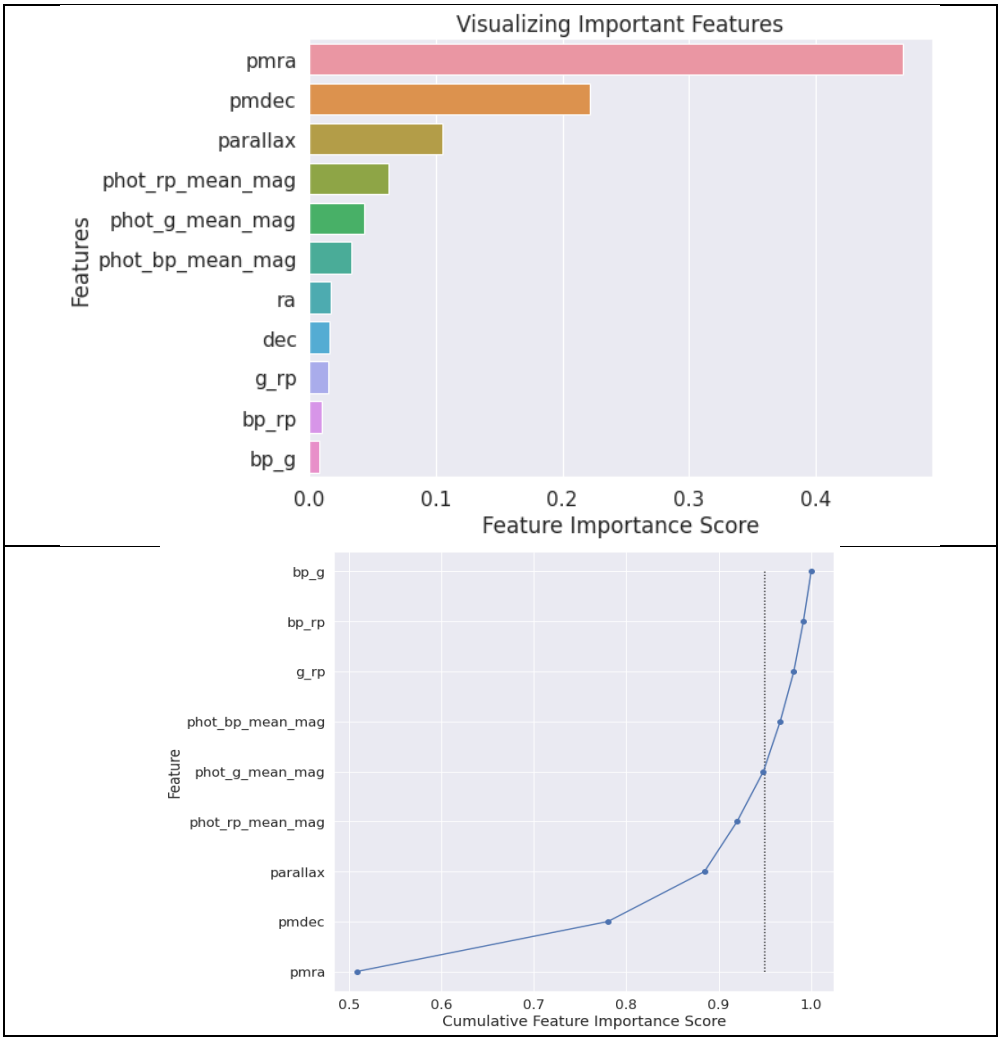


Fig 5.1: Feature importance of RF model using mean decrease in impurity (MDI). We want to only chose the parameters that are needed to cross a cumulative importance of 95% (shown by the dashed line in the lower plot). For M67, we selected only the best five important parameters.

For any metrics of feature importance, if they sum up to 1, we can use a threshold of 0.95 (95%) to select the important features. This tells us which set of features do we need to reach a relative feature importance score of 95%. to a We will calculate the cumulative feature importance of the features, ordered by their importance, (Fig 5.1., bottom plot) and choose the features below the 0.95 threshold line (dotted line in the plot).

One limitation of MDI is that it is strongly biased on the training data. It gives a good information on which features are most important in training data, but not necessarily provide good information on feature importance for the test datasets. Secondly, MDI favors the high cardinality features, the features with many unique values. Thus, a random numerical variable can get more scores in MDI (Breiman, 2001). Thirdly, MDI only works for tree-based models.

Another metric for feature importance, which solved these problems and can work with any type of supervised models, is called Permutation Feature Importance (PFI). It calculates the importance of a feature by calculating the decrease of the model score when that feature is randomly shuffled (Breiman, 2001). First it calculates the model base score, s_{base} based on a given metric (for example, precision for a classifier model) using all the features and samples. Then for each feature f, it randomly shuffles the feature and compute the model score, s_f again. Ideally, it repeats the process several times and calculates the average model score, $\bar{s_f}$ with randomly shuffled feature *f*. The PFI of feature *f* for the given model is defined as,

$$PFI_j = s_{base} - \bar{s_f}$$

The main idea behind PFI is that, if a feature is very important for the final prediction, then a random shuffle will break the relationships between the feature and the output, resulting in a large decrease in model score. But PFI cannot work well if there are two or more important features which are correlated to each other. Then when one of them is randomly shuffled, the other correlated feature can still predict the output well and vice versa. As a result, both (or all) of them shows a very small importance using PFI, while in truth they both (or all) are important predictors.

The most recent metric for feature importance in any ML model is SHAP (SHapley Additive exPlanations) introduced by (Lundberg & Lee, 2017). The idea of Shapley value comes directly from game theory, and it measures the average marginal contribution of a feature in a model. Suppose a model consists of three features *A,B* and *C* and the model prediction is denoted by $v_{\{A,B,C\}}$. When we add an additional feature in a model, the marginal contribution of that feature in the new model is defined as the change in model prediction. Suppose if we have a model with (A, B) with prediction $v_{\{A,B\}}$ and addition of *C* changes the prediction to $v_{\{A,B,C\}}$, then the marginal contribution of *C* in model (A,B,C) is $MC_{C,\{A,B,C\}} = v_{\{A,B,C\}} - v_{\{A,B\}}$. When there is no feature chosen, the model prediction is simply the mean of the training outputs, i.e., $v_{\phi} = \overline{y_{train}}$. Now to calculate the SHAP value for a feature *f* we need to take a weighted average of its marginal contribution in all the models where *f* is present. The weight will be determined by the probability of choosing a specific set of features. The total number of ways, we can choose a set of feature S among a total of N features is,

$$C(N, S) = \frac{N!}{(N - S)!\, S!}$$

Now the total number of ways we can choose $(S + 1)$ features among N features while fixing the last element as *f*:

$$N_{S+\{f\}} = \frac{N!}{(N - (S + 1))!\, S!} = \frac{N!}{(N - S - 1)!\, S!}$$

Note that in the denominator, we have $S!$ Instead of $(S + 1)!$ because the last element is fixed as *f*. The probability of finding such a set would be,

$$P_{S+\{f\}} = \frac{1}{N_{S+\{f\}}} = \frac{S!\,(N-S-1)!}{N!}$$

Therefore, the SHAP values for feature *f* would be,

$$SHAP_f = \sum_{S \subseteq N \setminus \{f\}} p_{S+\{f\}} \times MC_{f,\{S+\{f\}\}}$$

$$= \sum_{S \subseteq N \setminus \{f\}} \frac{|S|!\,(|N|-|S|-1)!}{|N|!} \times (v_{S \cup \{f\}} - v_S)$$

Here, we are summing over all possible set from the full set of features N, where *f* is not present. Inside of the summation, first term is the probability of finding set S union *f* and the second term calculates the marginal probability of feature *f* in the set S union *f* ($S \cup \{f\}$). $|S|$ and $|N|$ denotes the length of the set S and N respectively.

This is how SHAP can inform us about the global (i.e., across the whole dataset) feature importance. The final value of SHAP represents how larger a feature is contributing to the final output of the model. A positive SHAP value means the feature pushes the output further away from the mean output ($\overline{y_{train}}$: also denotes as the *base value*) and a feature with negative SHAP value decreases the output.

But SHAP can also help us to know local feature importance: how the features contribute to predict a final outcome for a specific test sample. When we run the model in test data, it can compare the value of each feature with the average value of that feature in the training data. Suppose feature *f* has a positive SHAP value and its average value in training data is $\bar{f}_{tr}$ and its value in a specific test sample is f_{sample}. For $f_{sample} > \overline{f_{tr}}$, this feature will contribute to increase the outcome from the base value and for $f_{sample} < \overline{f_{tr}}$, it will decrease the

outcome. On the contrary, if feature f had a negative SHAP values, then it will be negatively correlated, and the final prediction would decrease and increase from the base value for $f_{sample} > \overline{f_{tr}}$ and $f_{sample} < \overline{f_{tr}}$ respectively.

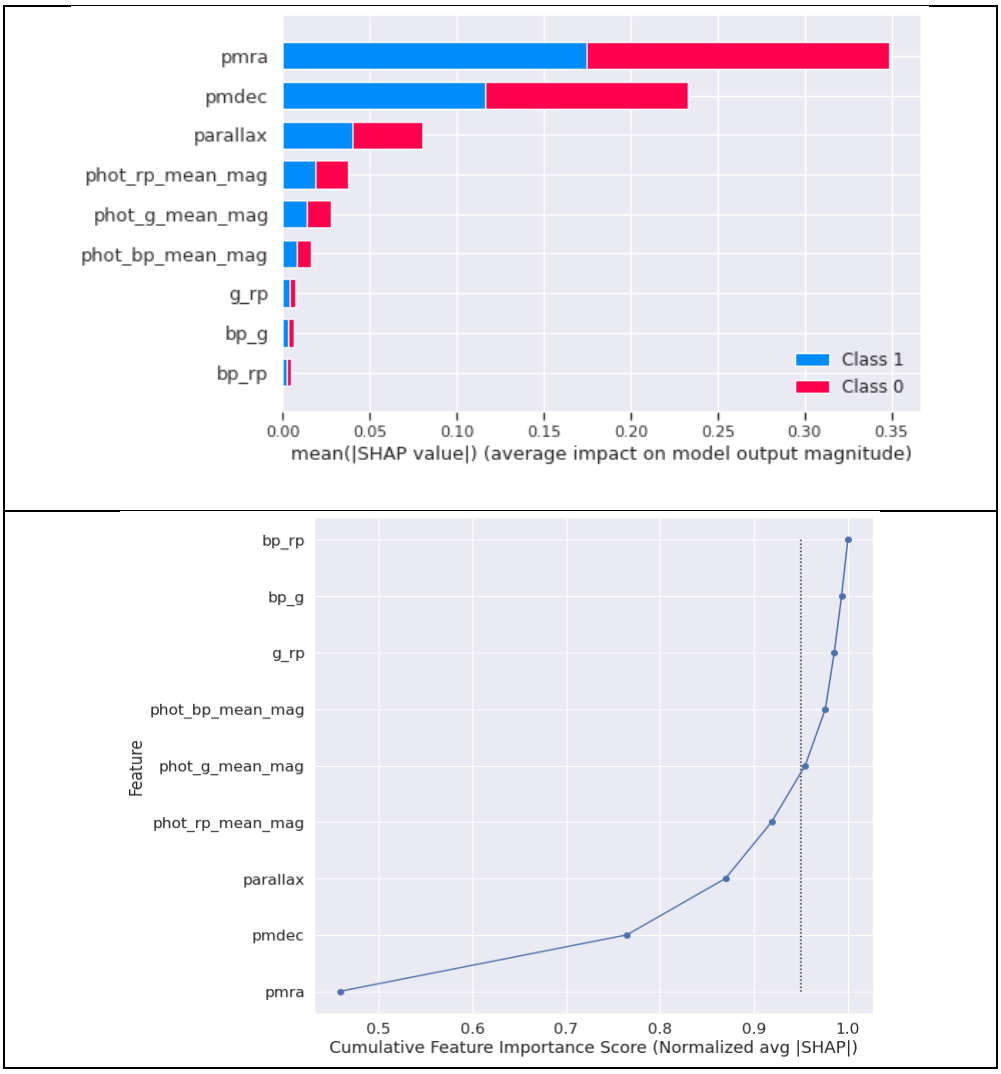


Fig 5.2: Feature importance of RF model using normalized average of absolute SHAP values for all the features. We want to only chose the parameters that are needed to cross a cumulative importance of 95% (shown by the dashed line in the lower plot). For M67, using SHAP values, we again found that only the best five important parameters are needed for 95% threshold.

In python's *shap* package, we can provide a fitted model and the features of a dataset. It will return all the local SHAP values for each of the features for all samples of the dataset. We can check the feature SHAP values for any specific sample for local interpretation. For global

interpretation, I can take the average of the absolute SHAP values of a feature across all the samples. If the output is a multiclass output, then we would get a different set of local SHAP values for each of the class. In our example, we have two output class: 1 (member) or 0 (non-member/field). Thus, we got two separate set of local SHAP values. Each set contains the local SHAP values of each feature for each of the samples. To calculate the importance of the features, we summed up the average of their absolute local SHAP values for each class. Fig 5.2 shows the feature importance in a decreasing order. Note that we did not take *ra* and *dec* as a feature because as we explained in section 5.1, our target region is larger than the stars in the training data in terms of *ra* and dec, and RF cannot do extrapolate. Thus, we want to train our model only with the other parameters. As expected, the astrometric variables (i.e., *pmra, pmdec, parallax*) played the most important roles to determine the outcome. To apply a threshold, we normalized the average of absolute SHAP values and choose the features which are needed to reach a cumulative 0.95 relative importance score.

5.3.2. Stratified Train-Test Split

For an unbiased measure of the performance of the model, we need a held-out test dataset, which we will not use in any part of the training. After the training finishes, we can use this test subset to understand how our model will perform to a real-world data. We will divide our full labelled training data, obtained from GMM model, into a training subset and test subset with an 80%-20% split.

Another important point is that our labelled data does not necessarily contain same number of member and field stars. If we do a random test-train split, there will be difference in the fraction of members between the training and test subsets. This, in result, can mislead our

analysis. To avoid this problem, we used a stratified test-train split method, where we ensured a similar distribution of member and field stars in both data subset.

5.3.3. Optimizing Model Parameters

Next, we need to choose and justify the values for the model hyperparameter. We can consider this as an optimization problem. Ideally, we want to choose a set of model hyperparameters that will ensure the best model, which we can measure by the performance metric. For supervised model, our chosen metric is precision, which will be the function we want to optimize (maximize). The parameters that we can change are the model hyperparameters. We want to reduce overfitting, so we will not be going to use same dataset for training and evaluating the model. But we also do not want to reduce the size of our training dataset. One way we can incorporate both idea is by cross-validation. In a k-fold cross validation, we split our data into k different set, then we train using k-1 different set each time and get the validation score using the held-out set. We continue to do that until we use all the sets as a validation set and then take the overall average score as the final performance score.

Now we can choose the hyperparameters by either doing a grid-search (make a grid of parameter and chose all of them) or random-search (randomly select parameter value everytime). We used random search instead of grid-search because it has two advantages (Bergstra & Bengio, 2012). One, we can set to do a finite number of iterations in a random search instead of completing all the grid points, which is computationally less costly. Two, if we sample randomly, there is a higher chance that we could explore the important part of the parameters more than a grid-search, specially when we have a large number of parameters. Usually cross-validation with random-search found the local maxima instead of global maxima because we are

not exploring all possible combinations. But if we run this for a very large number of times, then it will be more likely to reach to the global maxima.

N_estimators (Number of decision trees)	500, 1000, 1500, 2000, 2500, 3000, 3500, 4000, 4500, 5000, 5500, 6000, 6500, 7000, 7500, 8000, 8500, 9000, 9500, 10000
max_features (Number of features in each tree)	'sqrt', 1, 2, 3, 4, …, n_feature
max_depth (Maximum depth of the tree)	10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, None
min_samples_leaf (Minimum samples required for a leaf node)	1, 2, 4
min_samples_split (Minimum samples required to split a node)	2, 5, 10

Table 5.2: The parameter space from where we took the random samples for the randomized CV cross validation.

We used the grid presented in Table 5.2 to randomly select 100 models with different model parameter values to build a 5-fold cross validation model. Then we chose the best model among those 500 iterations. The best parameters found in this process are compared with the Gao paper's model parameter in Table 5.3.

Model Parameters	Gao Paper (2018a)	Modified Replication (Optimized by cross validation)
n_estimators (Number of decision trees)	7000 (chosen)	5500
max_features (Number of features in each tree)	4 (chosen)	4
max_depth (Maximum depth of the tree)	None (default)	70
min_samples_leaf (Minimum samples required for a leaf node)	2 (default)	2
min_samples_split (Minimum samples required to split a node)	1 (default)	2

Table 5.3: The model parameters in Gao (2018a) paper and the optimized parameters found using cross validation.

Now after justifying the feature selection, test-train split and hyperparameter optimization, our RF model is ready to be applied in a new target dataset. Similar to Gao (2018a), we search and collect all the stars for a larger radius (2.5 degree) around the cluster center. One important consideration in data preprocessing is that RF model cannot do extrapolation. Any cutoff, that have been applied to the features of the training data, have to be applied also for the target stars. Our training data comes from GMM model, where we initially applied a proper motion cutoff to avoid outlier (*pmra* and *pmdec* should be between -20 to 20 mas/yr) and a distance cutoff chosen by the MSS metric for ensuring GMM model performance. We need to apply those similar cutoffs to the target stars, which significantly reduced the number of targe stars.

The overall result from the modified replication is presented in Table 5.3. Even with a less than half of the target stars compared to Gao paper, the number of predicted member stars increased in this model slightly in addition to the stronger justifications of the model. All the figures still show the similar result was found by Gao.

	Gao M67 Paper (2018a)	Modified Replication
Target Stars	71,117	30,069
n(PMemb >= 0.6)	1502	1489
n(PMemb >= 0.8)	1361	1447

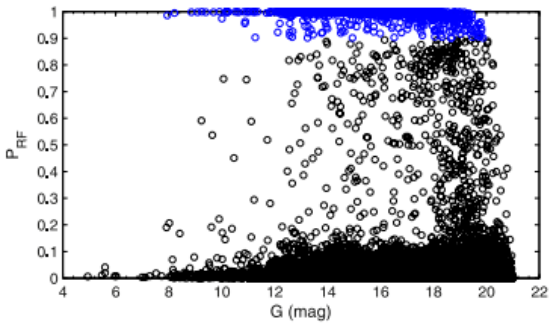
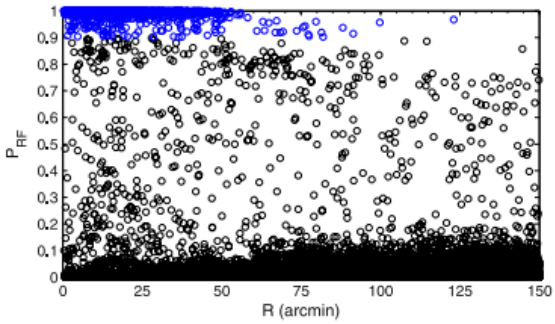
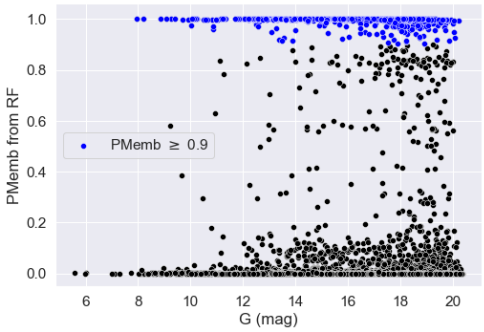
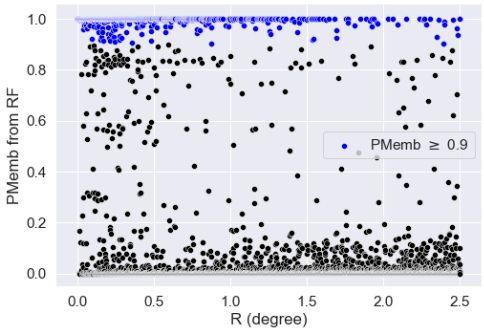
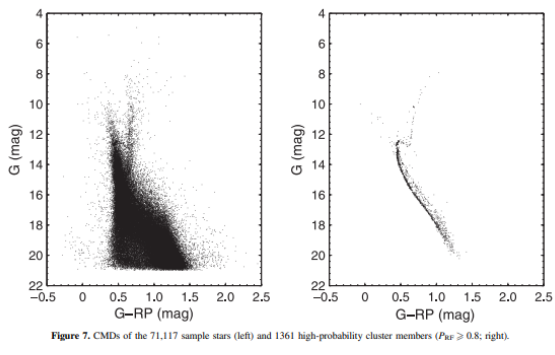
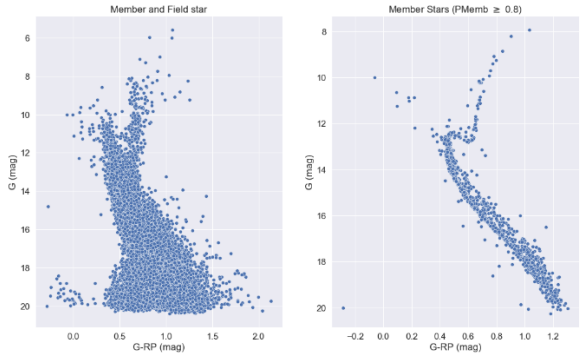
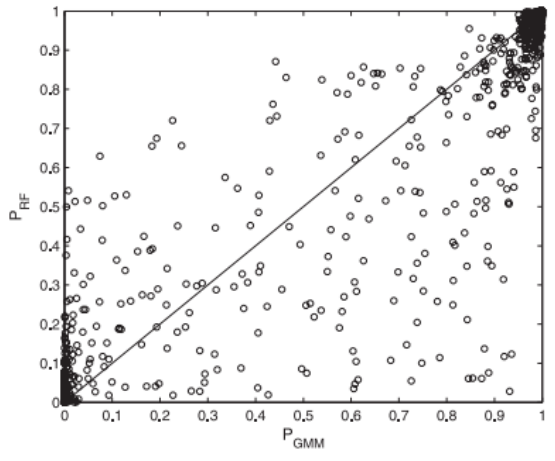
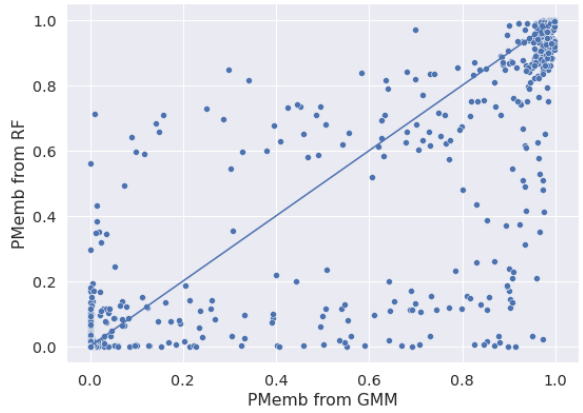
PMemb vs G Mag, PMemb vs R (distance from the cluster center)	 	 
CMD	 **Figure 7.** CMDs of the 71,117 sample stars (left) and 1361 high-probability cluster members ($P_{RF} \geq 0.8$; right).	
Comparison between the PMemb of both models.		

Table 5.4: Comparison between the modified replication and the original paper.

5.4. Running RF model in a new cluster

Lastly, we applied the similar procedures explained in modified replication in NGC 3766 using the GMM output as the training data. First, we use normalized version of the average absolute SHAP values to select the most important features. Fig 5.3 shows the SHAP feature importance and the cumulative feature importance of the features.

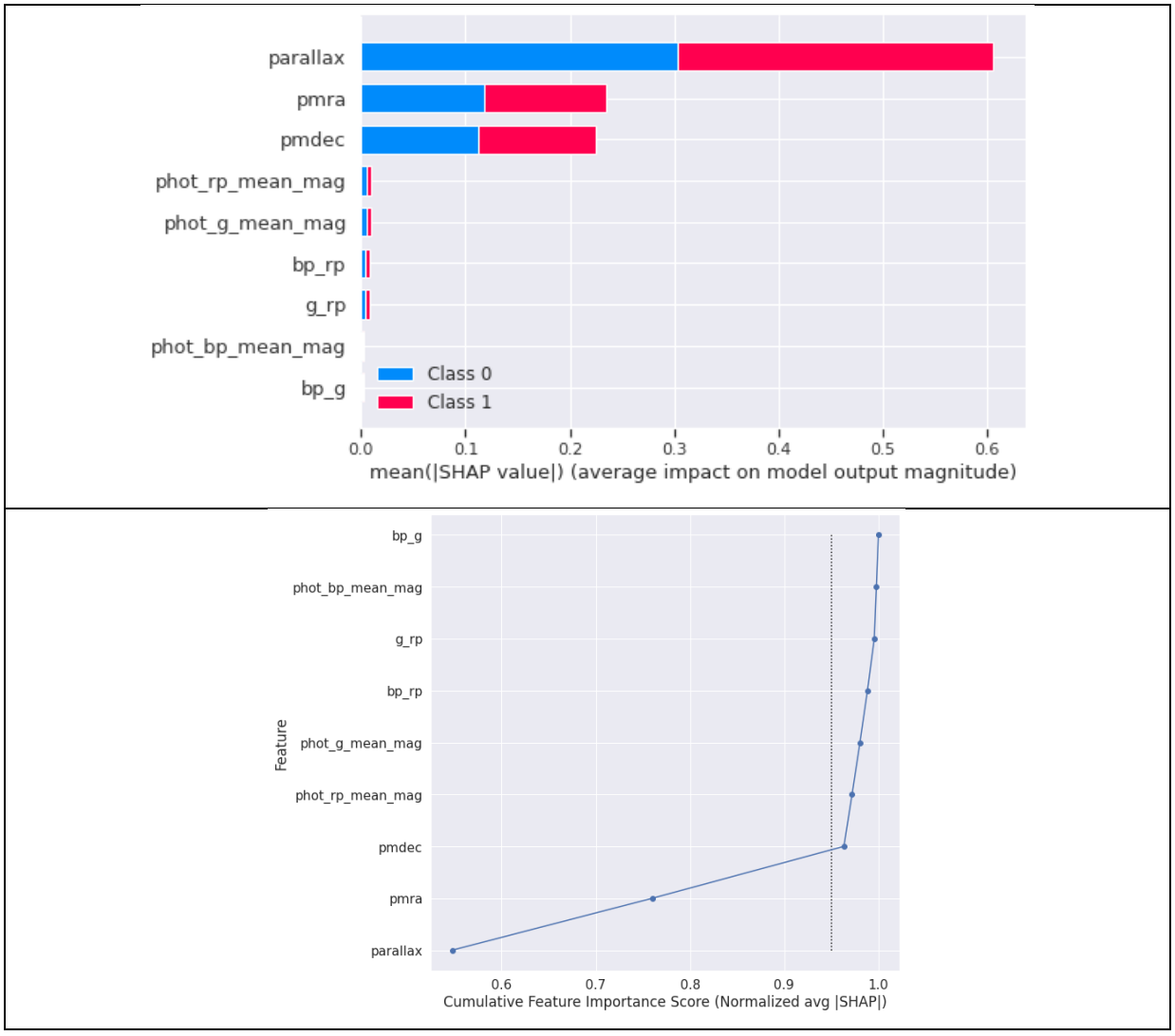


Fig 5.4: Feature importance of RF model using normalized average of absolute SHAP values for all the features. We want to only chose the parameters that are needed to cross a cumulative importance of 95% (shown by the dashed line in the lower plot). For NGC 3766, using SHAP values, we found that only the best three important parameters are needed for 95% threshold

Model Parameters	NGC 3766 (Optimized by cross validation)
n_estimators (Number of decision trees)	5500
max_features (Number of features in each tree)	*sqrt(n_features)*
max_depth (Maximum depth of the tree)	110
min_samples_leaf (Minimum samples required for a leaf node)	2
min_samples_split (Minimum samples required to split a node)	2

Table 5.5: The optimized model parameters for NGC 3766 found using cross validation.

Based on 95% threshold, we only found *pmra*, *pmdec* and *parallax* as the important parameters and used them in RF model. Next, to optimize the hyperparameter, we applied a 5-fold 50 iterations cross-validation. As the number of sample stars are larger, we used a smaller number of iterations to reduce the runtime. We get a precision score of 1.0 in the validation subset. If we get a smaller precision, we will consider running the model for longer iterations to improve the performance. The found optimized hyperparameters are listed in Table 5.5.

Then we selected all the stars from a larger region (1 degree radius) around the cluster center. Similar to the modified replication, we applied the same proper motion and distance cutoff of the training data to the target data. Finally, we applied RF to this new filtered data and the results of the RF model are summarized in Table 5.6. When we plot the CMD of the stars that have more than 0.8 PMemb by RF model, it still looks very similar to the CMD of all stars. So, we applied the strictest cutoff and only select the stars with PMemb = 1 as our member stars.

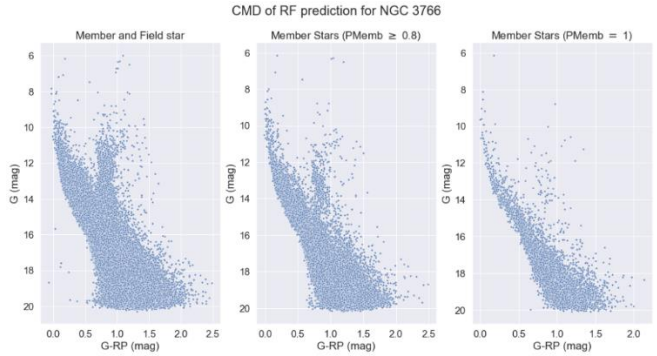
Target Stars	1,71,025
PMemb >= 0.8	50,009
PMemb = 1	6,352
CMD	

Table 5.6: Results of the RF model applied for the NGC 3766 open cluster.

6. Results

6.1. Distribution of Parameters

If we compare the results between all these analyses, we can see that our modified approach significantly improve the number of members. All models have a very high precision and the similar precision in train and test data shows that there is likely no overfitting. But high precision will mean a good model only if the MSS score for GMM is also good. We can see that using a 0.95 member cutoff, only the modified replication of M67 shows more than 0.91 MSS value. For the new cluster, even using the 0.95 member cutoff, MSS value only reaches to 0.78, though it improves significantly from the performance with 0.6 cutoff (MSS = 0.61).

	Direct Replication M67	Modified Replication M67	NGC 3766
MSS (GMM) @ 0.6	0.70	0.89	0.61
MSS (GMM) @ 0.95	0.73	0.91	0.78
Precision (RF) in train data	1.00	1.00	1.00
Precision (RF) in test data	-	1.00	1.00
Number of Members	1375 (0.8)	1447 (0.8)	6352 (1)

Table 6.1: Comparison between the results of all three analysis

Fig 6.1 shows that all three astrometric parameters (*pmra, pmdec, parallax)* of M67 and NGC 3766 are strongly normally distributed, which supports the assumption of normally distributed members and uniformly distributed field stars in a open cluster region. The CMD of M67 clearly shows the turn-off point and closely follows a theoretical CMD. But the younger cluster NGC 3766 shows a very broad CMD, which can be an indicator that some predicted members may not be a true member.

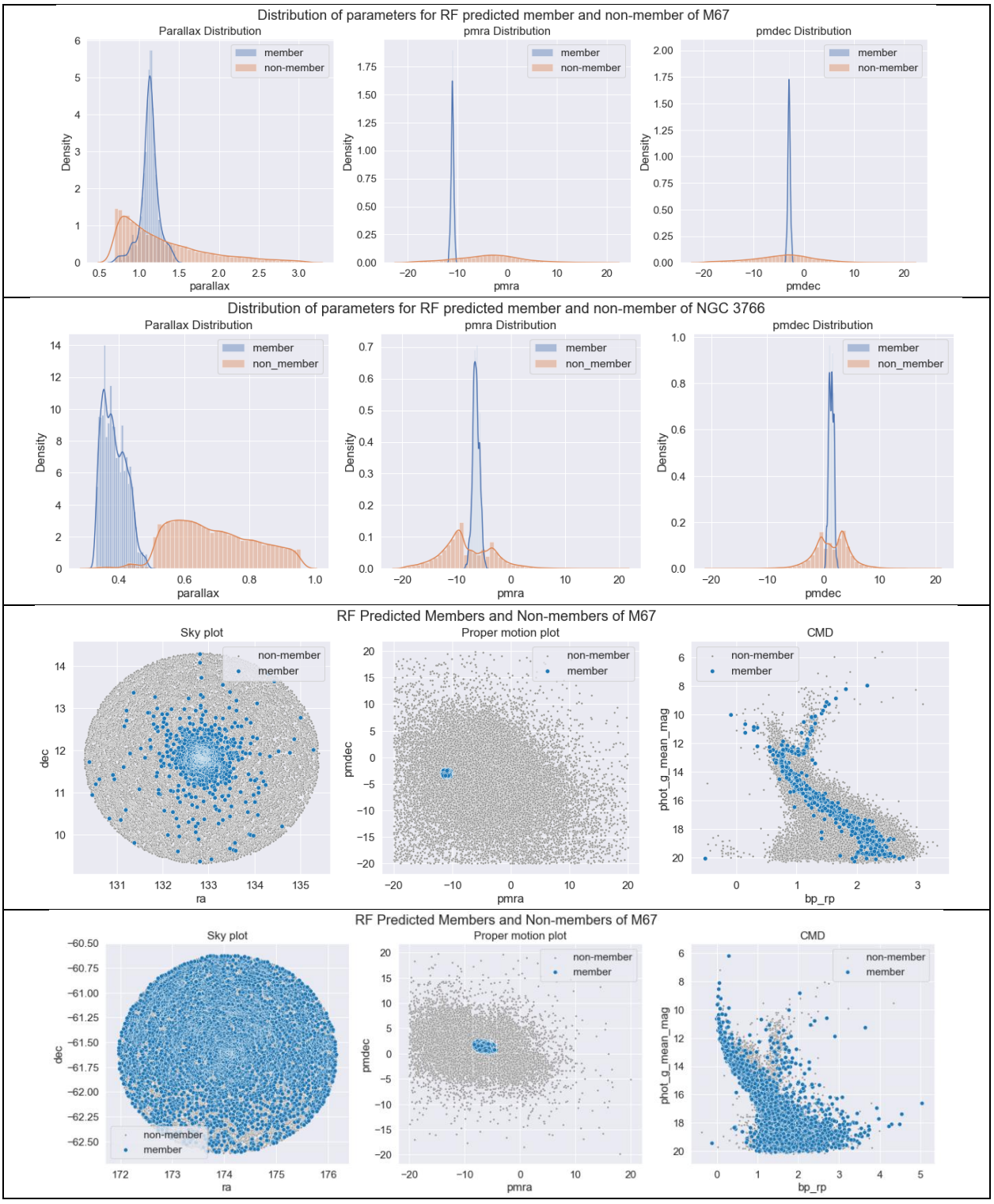


Figure 6.1: Distribution of parameters for both M67 and NGC3766. For both cluster the member parameters are distributed normally with a very low standard deviation, where the field stars are very broad and uniform. In the proper motion space, again the member stars are very compact. But M67 shows a better result in their CMD and sky plot. The CMD of NGC 3766 predicted members are much deviated from the theoretical CMD.

6.2. Compare with the Cantat Benchmark

When we compare our result with Cantat-Gaudin et al. (2020) dataset, we can see that the number of member increases and most of the additional members are found at the outer sky region and fainter magnitudes.

Number of Members	M67	NGC 3766
Cantat-Gaudin paper (2020)	845	1368
Our Analysis (Modified Version)	1447	6352

Table 6.2: Comparison between the number of members found by Cantat (2020) and our analysis

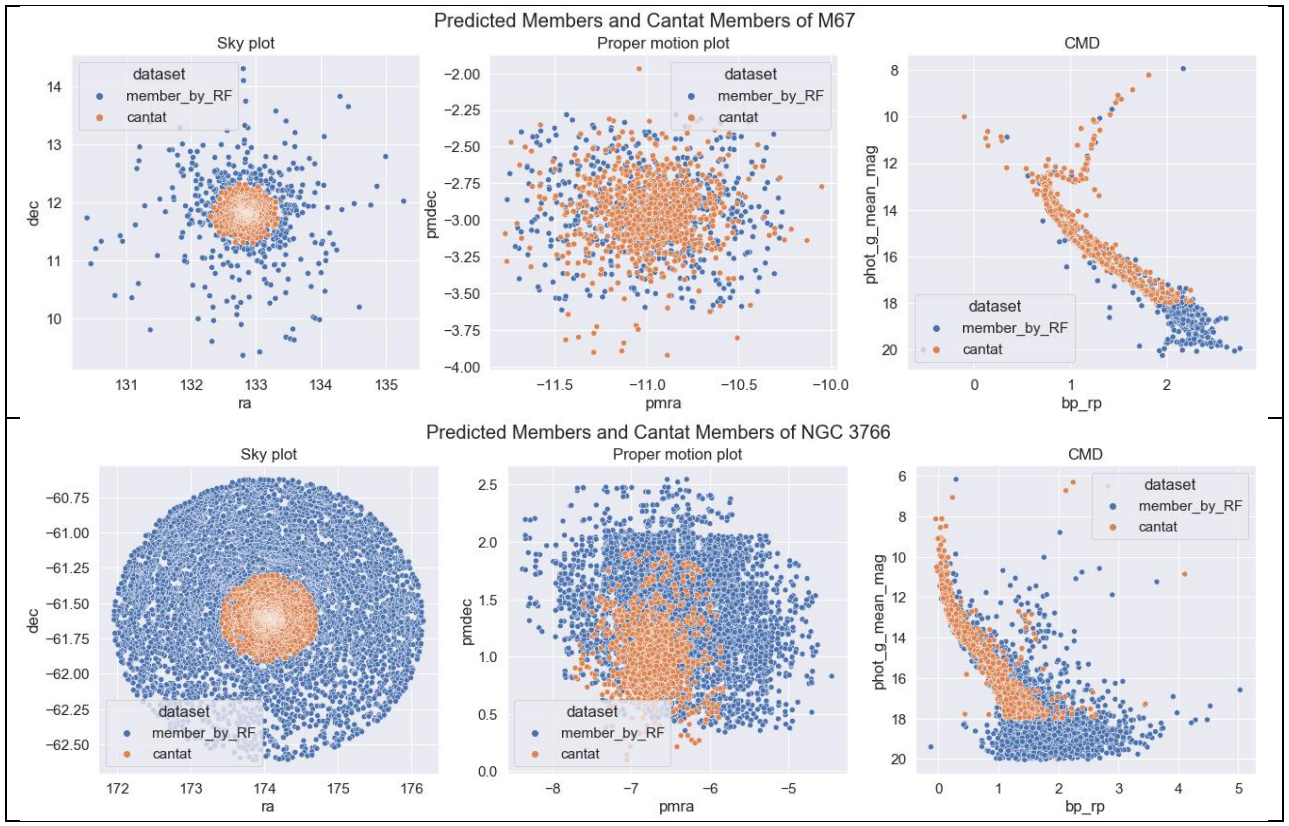


Figure 6.2: Comparison between Cantat data and our analysis shows that our analysis mostly captured all the members captured by Cantat. But additionally in the outer region of the cluster and the lower part of the CMD (fainter region), we have more members found in our study

7. Conclusion and Future Work

In this paper, we worked with the hypothesis that a well-justified and optimized semi-supervised model would be a better approach compared to an unsupervised model to detect members of open star clusters in terms of both model performance and retrieved members. The majority of the paper focused on developing the strategy to find the best semi-supervised model.

First, we need to have a numerical metric to compare several candidate unsupervised models and another similar metric for supervised models. I proposed a new metric *Modified Silhouette Score (MSS),* to compare unsupervised models using only their predicted member and field star group. Using a simulated dataset, I empirically showed how MSS could be a good metric to measure the model performance to distinguish normally distributed members from the uniformly distributed field stars. For comparing supervised models, I chose precision as the evaluation metric to reduce false positives (field stars identified as a member).

Before comparing the models with the chosen metrics, we need to ensure that all our models are well-justified and optimized. In this paper, I thoroughly discussed GMM and RF to show how we can justify and optimize our unsupervised and supervised model.

For an unsupervised model, first, we need to understand the underlying assumptions of the model. For example, as GMM assumed a normal distribution of both groups, we could not use the photometric variables as they are not normally distributed for members. Secondly, if necessary, we can use the MSS metric for further feature selection. We can test with different combinations of features and choose the combination which results in the best MSS value. Similarly, MSS can also be used to optimize the value of any model hyperparameters (for example, k in k-means clustering) or any problem-specific parameter (for example, half-width of distance in GMM). Most of the unsupervised models would just output two different groups. If

any model provides a membership probability (like GMM), we need to apply a member cutoff to choose the most reliable members and most reliable field stars. Again, one strategy could be to use MSS to choose the best member cutoff.

We should also take into consideration the model-specific limitations. For example, one major drawback of GMM that we discussed in section 4.2 is that it cannot distinguish between two groups if they have a similar peak (i.e., mode) in the distribution of their features. This is why we cannot use *ra* and *dec* as a feature in GMM, which means we are unable to use some precious information. Similarly, if for any star cluster, the member and field star distributions in any of the *parallax, pmra,* and *pmdec* peaks at the same value, then GMM will perform very poorly to determine the members.

For the supervised model, again, we need to first identify and analyze the underlying assumptions behind the algorithm. Then we can use SHAP feature importance to select the most important features. One strategy could be to only take those features that are needed to cross a 95% cumulative feature importance. As the training data is coming directly from an unsupervised model, most likely, it will be imbalanced, i.e., the number of members and field stars will be different. Thus, it is important to always use a stratified test-train split to divide the data into test and training sets. Lastly, for optimizing the model parameters, we should use cross-validation using the precision (or any other justified choice of metric).

Similar to the unsupervised model, we should always consider the limitations of any supervised model. For example, one limitation of RF and other decision tree-based algorithms is that they cannot do extrapolation. Thus, if we apply any filter in the training features, we need to apply the same filters in the target dataset.

Regarding the evaluation metric, MSS is just one choice and by no means perfect. One limitation of MSS is that if the predicted member group is very small (4-5 stars only), its standard deviation becomes very low, and the numerator of MSS becomes approximately the standard deviation of the field star group ($SD_{field} - SD_{member} \approx SD_{field}$). Then irrespective of the actual distribution of the member and field star, the MSS value becomes close to 1. Similarly, when we have a very large field star group, thus a large SD_{field}, again the numerator equals approximately SD_{field} and MSS becomes close to 1, irrespective of the member distribution. But as shown in Figure 3.3, in most cases, MSS measures the model performance correctly. So, we can say that though there could be many ways to improve the MSS metric; nevertheless, it can be a very useful metric in this context.

Similarly, for comparing supervised models, one may consider more than one metric. In addition to the precision, one could also use the recall score or the area under the ROC curve to consider both false positive and negative. But we always need to justify the choice of the metrics with solid reasoning.

Once we find the best semi-supervised model, we need to choose and specify which unsupervised model we should take to compare with our model. One of the best options could be to take the current benchmark paper on the all-sky open star cluster member catalog: Cantat-Gaudin et al. (2020) paper. In order to use the MSS metric, we need to have both predicted member and field star group. But Cantat-Gaudin et al. (2020) only reported their member group in their paper. Thus, in order to use this paper, we need to replicate the unsupervised model used by the paper to get both the members and field stars.

Now that we have both our semi-supervised model and an unsupervised benchmark model, we can run them for a number of star clusters. Then we will compare the number of retrieved members in each model. Secondly, we will compare the model performance using the chosen metrics. For the Cantat-Gaudin et al. (2020) paper, we will use the MSS score. But for our semi-supervised model, we have two metrics: one for unsupervised (MSS) and one for supervised (precision). High performance in supervised model metrics does not necessarily imply a better model. A model with high precision only means that it can accurately map the training data. Thus, its overall performance is dependent on the confidence of the unsupervised model. One way we can incorporate both metrics is to multiply the MSS with the precision score. Then to get an overall good performance score, one has to be confident in both unsupervised and supervised models.

Finally, in order to support our hypothesis, we need to consider both the number of members and the overall performance of our semi-supervised model compared to the Cantat-Gaudin paper. Our hypothesis will be supported if we get the following results. The best case is if we detect more members with more confidence, i.e., both the number of members and the performance metrics are larger for our model. Similarly, if performance metrics for both models are very similar, but we get more members in the semi-supervised model, it also supports the argument as we increased the number of confident members. If this hypothesis is true, next, we can try to apply this model to all identified open star clusters and analyze if the chosen optimal unsupervised and supervised ML models stay the same for all clusters or not. For example, we can check if GMM is better for older clusters (where parameters should be more compact, thus more normally distributed) than the younger clusters.

If we find that the model performance is higher for the semi-supervised model, but we have an almost similar or fewer number of members, it will suggest that the hypothesis is not true in its entirety. Only part of the hypothesis can be true that having a semi-supervised model increases our confidence in the predicted members.

Lastly, for the following cases, we can conclude that the hypothesis is more likely to be false. The strongest evidence for falsifying the hypothesis will be the decrease in both the number of members and the model performance in our model. Similarly, if the model performance stays the same and the number of members stays decreases (or even stays the same), then the semi-supervised model is not so helpful. It just adds more complexity cost without any significant improvement. If the model performance is lower, then even an increase of members does not support the hypothesis, as we will not be confident enough for these additional members. In this case, as we find that an unsupervised model could be very optimal in this context, our next step could be to focus on developing a more improved common quantitative metric for unsupervised models building upon MSS.

Appendix: Code Notebook

GitHub Repository:

https://github.com/mahmud-nobe/Minerva-Capstone

1. Understanding GMM:

Here, I used simulated data to test how the GMM performances changes for varying half-width value for different number of stars (explained in Section 4.3). Additionally, in the appendix, I showed how the MSS metric goes from smaller to larger for GMM performance in different simulated dataset. (Section 3.3)

2. *Direct Replication M67*:

Here, I directly replicated Gao (2018a) paper to apply GMM and RF for open cluster M67 (explained in Section 4.4, 5.2).

3. *Modified Replication M67:*

Here, I applied the necessary modification to improve the justification of both GMM and RF model for open cluster M67 (section 4.5, 5.3).

4. *New Cluster Modified Model*:

Here, I applied the strategy developed in the modified replication to both GMM and RF model and apply it to a new cluster NGC 3766.

Code Snippets: Class for simulating dataset (Used in notebook 1)

```python
class Simulated_Data():
    '''
    This class creates the simulated member and non-member dataset for a given gridsize.
    The members are normally distributed and the non-members are uniformly distributed.

    Attribute:
    ----------
    size:        Gridsize of the simulation
    center_x:    x coordinate of member group's center (at the center of grid)
    center_y:    y coordinate of member group's center (at the center of grid)

    n_field:     Number of field (non-member) stars
    n_member:    Number of member stars
    std_member:  Standard deviation of the member group normal distribution
    data:        Simulated data:
                    Columns: feature 1, feature 2 and True label (1: member, 0: non-member)
    '''
    def __init__(self, size = 20):
        self.size = size  # size of the full grid
        self.center_x = size/2
        self.center_y = size/2

    def initialize(self, n_field, n_member = 200, std_member = 1):
        '''
        Initialize the simulated dataset for the given number of member and field star.

        Input:
        ------
        n_field:     Number of field (non-member) stars
        n_member:    Number of member stars
        std_member:  Standard deviation of the member group normal distribution
        '''
        self.n_field = n_field
        self.n_member = n_member
        self.std_member = std_member # standard deviation of members

        # normally distributed members
        mem_feature_1 = sts.norm(self.center_x, self.std_member).rvs(self.n_member)
        mem_feature_2 = sts.norm(self.center_y, self.std_member).rvs(self.n_member)

        # uniformly distributed field stars
        field_feature_1 = sts.uniform(0,self.size).rvs(self.n_field)
        field_feature_2 = sts.uniform(0,self.size).rvs(self.n_field)

        # create the dataframe for simulated data
        self.data = pd.DataFrame({'star': range(1,self.n_member+self.n_field+1),
                     "feature_1": np.append(mem_feature_1, field_feature_1),
                     'feature_2': np.append(mem_feature_2, field_feature_2),
                     'true_label': [1]*n_member+[0]*n_field})

        # Any stars that are closer than 2*SD from the center are categorized as member star
        mask = np.sqrt((self.data.feature_1 - self.center_x)**2 +\
                       (self.data.feature_2 - self.center_y)**2) < 2*self.std_member
        self.data.loc[mask, 'true_label'] = 1

    def visualize(self, title = None):
        '''Visualizes the full simulated dataset'''

        sns.scatterplot(data = self.data, x='feature_1', y='feature_2', hue = 'true_label',
                        palette = 'viridis')
        plt.xlabel('Feature 1')
        plt.ylabel('Feature 2')
        plt.legend(title = 'True Label')
        plt.title(title)
        plt.show()
```

Code Snippets: Class for running GMM model (Used in notebook 3 and 4)

[15]:

```
class Run_GMM_Model():
    '''
    This class runs GMM model to the selected part of the datset, predicts the member and field star
    group, calculates the evaluation metric and visualizes the results

    Attribute:
    ----------
    data:           Full dataset of stars (after applying noise filter)
    working_data:   Filtered dataset by a given half-width value.
                    We will run GMM model in this filtered dataset

    (The following attribute is used to determine the working_data)
    distance_lit:   Distance of the cluster from the literature

    feature_colums: The list of columns used as the features of GMM model
    gmm:            The GMM model applied on working data
    member:         The predicted member stars by GMM model in working data
    non_member:     The predicted field stars by GMM model in working data
    mss_metric:     Modified Silhouttee Score (MSS) for the current member and non_member group
    '''

    def __init__(self, data, distance):
        '''
        Initializes the class with a dataset.

        Input:
        -------
        data:       Full dataset of stars
        distance:   Distance of the cluster from the literature
        '''
        self.data = data
        self.distance_lit = distance

    def get_working_data(self, half_width):
        '''
        Returns the filtered subset of the datset based on the given half-width

        Input:
        ------
        half_width: The half-width for the distance cutoff
        '''
        filter = abs(self.data.distance_pc - self.distance_lit) < half_width
        working_data = self.data.loc[filter, :]
        return working_data

    def get_member(self, half_width, feature_columns, cutoff = 0.6, random_state = None):
        '''
        Classifies the working data into member and non_member groups.

        Inputs:
        -------
        half_width:     The half-width for the distance cutoff
        feature_colums: The list of columns used as the features of GMM model
        cutoff:         Threshold used for member cutoff. Default is 0.6.
                        If cutoff is 0.6, stars with membership probability >= 0.6 are members
                        and the stars with membership probability <= 0.4 (1-0.6) are non-members
        random_state:   Random State of the GMM algorithm. Used for reproducibility.
                        If 'None', then each time a new random seed will be used. Default is None.

        GMM clusters the data into two groups. The group which has smaller average standard deviation
        (thus more compact) is defined as the member group.
        '''
        self.feature_columns = feature_columns
        self.working_data = self.get_working_data(half_width)

        features = self.working_data.loc[:,feature_columns].dropna()

        # if there is less than 2 stars, GMM cannot divide them in two groups
        if len(self.working_data) < 2:
            raise ValueError('Less than two stars in the data')

```

```
        # normalizing the features
        scaled_features = pd.DataFrame({})
        for column in features.columns:
            scaled_features[column] = (features[column] - np.median(features[column]))/np.std(feature

        # Running GMM model: n_init = Number of Different Initialization tried by GMM
        gmm = GaussianMixture(n_components=2, n_init = 5, random_state = random_state)
        gmm.fit(scaled_features)

        #predictions from gmm
        labels = gmm.predict(scaled_features)       # group no (0 or 1)
        probs = gmm.predict_proba(scaled_features)  # membership probability

        # first assuming the group 1 is the member group
        self.working_data.loc[:, 'PMemb'] = probs[:, 1]  # membership probability to be in group 1
        self.working_data.loc[:, 'gmm_label'] = labels

        # select member and non-member based on member threshold
        non_member_ind = self.working_data.loc[:, 'PMemb'] <= (1-cutoff)
        non_member = self.working_data.loc[non_member_ind, :]

        member_ind = self.working_data.loc[:, 'PMemb'] >= cutoff
        member = self.working_data.loc[member_ind, :]

        # check if the average standard deviation of member group is larger.
        # if yes, then change the member group and assign group 0 as the member group.
        if member[feature_columns].std().mean() > non_member[feature_columns].std().mean():
            self.working_data.loc[:, 'PMemb'] = probs[:, 0] # membership probability to be in group 1
            self.working_data.loc[:, 'gmm_label'] = 1-labels

            # select member and non-member based on member threshold using new PMemb value
            non_member_ind = self.working_data.loc[:, 'PMemb'] <= (1-cutoff)
            non_member = self.working_data.loc[non_member_ind, :]

            member_ind = self.working_data.loc[:, 'PMemb'] >= cutoff
            member = self.working_data.loc[member_ind, :]

        # save the results as an attribute of the class
        self.member, self.non_member, self.gmm = member, non_member, gmm

    def get_MSS_metric(self, epsilon = 1e-7):
        '''
        Calculates modified silhouttee score (MSS) from the current member and non_member group.
        If any of the group has less than 2 member (thus SD = 0 or None), MSS value is 0.
        Otherwise, MSS is the average of (field_SD - member SD)/max(field_SD, member SD, epsilon) for

        epsilon: a very small number to avoid dividing by 0, when both SD are 0. Default value 1e-7.
        '''
        if len(self.member) < 2 or len(self.non_member) < 2:
            return 0

        metric = np.zeros(len(self.feature_columns))
        for i in range(len(self.feature_columns)):
            feature_i = self.feature_columns[i]
            # (field_SD - member SD)/max(field_SD, member SD, epsilon) for each features
            metric[i] = (np.std(self.non_member[feature_i]) - np.std(self.member[feature_i])) \
                    / max(np.std(self.member[feature_i]), np.std(self.non_member[feature_i]), epsilon

        self.mss_metric = metric.mean()
        return self.mss_metric

    def visualize_member(self, title = None):
        '''
        Visualizes the following four plots in a 2x2 (row x col) setup:
        1. Proper motion plot (pmra vs pmdec) for current member and non-member
        2. Parallax distribution for current member and non-member
        3. Color-Magnitude Diagram (CMD) for member group
        4. Color-Magnitude Diagram (CMD) for non-member group

        Input:
        -------
        title: An overall title of these set of figures
```

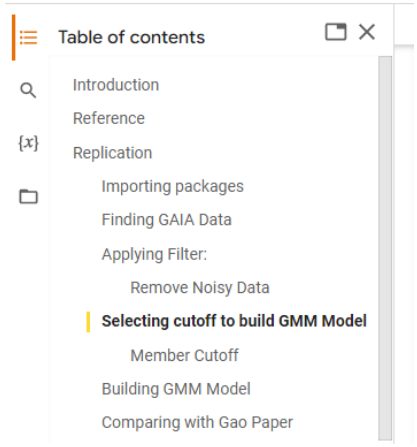
```
        '''
        plt.figure(figsize=(14,14))

        if title:
            plt.suptitle(title)

        plt.subplot(221)
        sns.scatterplot(data = self.non_member, x='pmra', y='pmdec',
                        label = 'non_member', color = 'tab:blue')
        sns.scatterplot(data = self.member, x='pmra', y='pmdec',
                        label = 'member', color = 'tab:red')
        plt.xlabel('pmra (mas/yr)')
        plt.ylabel('pmdec (mas/yr)')

        plt.subplot(222)
        sns.distplot(self.non_member.parallax, label = 'non_member',
                     color = 'tab:blue', kde = True)
        sns.distplot(self.member.parallax, label = 'member',
                     color = 'tab:red', kde = True)
        plt.xlabel('Parallax (mas)')
        plt.ylabel('Probability Density [$mas^{-1}$]')
        plt.legend()

        plt.subplot(223)
        sns.scatterplot(data = self.member, x='g_rp', y='phot_g_mean_mag',
                        color = 'tab:red', label = 'member')
        plt.gca().invert_yaxis()
        plt.ylabel('G Magnitude')
        plt.xlabel('G - RP Color')

        plt.subplot(224)
        sns.scatterplot(data = self.non_member, x='g_rp', y='phot_g_mean_mag',
                        color = 'tab:blue', label = 'non member')
        plt.gca().invert_yaxis()
        plt.ylabel('G Magnitude')
        plt.xlabel('G - RP Color')

        plt.show()
```

HC and LO Justification

LO Appendix

LO Rubrics can be found here: LO Rubrics - Mahmud

LOs	Place of application	Justification
#CS110-Python-Programming	Code Notebooks	For this project, I worked with four different notebooks, their name and brief summaries are mentioned in the code notebook appendix. Whenever necessary, we created well documented classes and functions to avoid repetition in python implementation. Two major class (*Simulated_Data* and *Run_GMM_model*) that have been implemented in python notebooks are shown in the above code snippet. Throughout the process, I followed the best practices of python coding to increase efficiency or reducing the time complexity. I used NumPy array and vectorization for most of the numeric calculation. Some of the good examples of using python are the followings. In simulated data, we want to create different simulation with varying half-width and different number of field stars. As we want to change the number of stars, I did not use number of clusters or field star in the *__init__* method. Instead, we created a different method called *initialize* in our *Simulated_Data* class. Now we only need to create only one instance of *Simulated_Data* object defining the grid size. Then we can use *initialize* method every time we want to create a new dataset with different number of field and clusters. To calculate the best half-width cutoff, I run the simulation with

		different half-width value. To confound for any random error, I run 15 different trials for a given half-width value and take the average. As we want the randomness, when finding the best half-width and best cutoff using the *get_member* method, I used *random_state=None*. But when I build my final model using the best cutoff and best half-width, I used a *random_state=42* for reproducibility. Another small point is while calculating the MSS metric in *get_MSS_metric*, I used a very small value epsilon in the denominator to avoid division by zero.
#CS110-Code-Readability	Code Notebooks	I followed the PEP 8 guidelines for writing codes for better readability. The codes are organized using class and function that makes the code modular and helps external readers to easily navigate the code. I also added short introduction at the beginning and divided the code under several headings:  I added efficient and useful docstring for all the class, functions, and the methods of the class (as shown in the code snippet). The class names start with CapWords convention (first letter of the word is uppercase) while the

		functions and methods start with lowercase letter. The name of the variables and the functions are meaningful and when appropriate verb is used for function and method name. In *get_member* method, we are using a GMM model with two components. It will fail to run if there are less than 2 stars in our dataset. So, I am raising a ValueError if the given dataset has less than 2 stars (line 68-70 of *Run_GMM_model)*. The use of indention is also constant throughout the notebooks, and they followed PEP 8 convention. For example, additional line(s) of a function argument aligns with the opening parenthesis:
		`161 sns.distplot(self.member.parallax, label = 'member',`
		`162 color = 'tab:red', kde = True)`

#NS125-DataPrep	Code Notebooks	For notebook 2,3 and 4, the GAIA data has been downloaded directly from the archive using *astropy* and *astroquery* packages. From the astropy table they are converted to *pandas* DataFrame for ease of analysis. The most important variables for our analysis are the photometric and astrometric variables. We cannot work with any star which is missing these columns; thus, we dropped any such observations. The magnitude error is calculated using the flux error given in the dataset, which is used to filter out more noisy data.
		`1 ## calculate magnitude error`
		`2 ## del magnitude = - 2.5 log(del Flux / Flux)`
		`3 all_stars['g_mag_error'] = 2.5/np.log(10) / all_stars.phot_g_mean_flux_over_error`
		Similarly, the distance is calculated from the parallax. To remove any outlier, we take only stars with positive parallax value and a proper motion range of -20 to 20 milli arcsec/year. Once we get the member and field star

		groups from GMM model, we merged them together to create the training data for RF. Similarly, to compare with the Cantat-Gaudin et al. (2020), we downloaded the Cantat data from the archive using *astropy*, then I merged Cantat data with our data using the GAIA *source id*, the common column in both dataset:
		`# renaming the cantat table to match it with gaia_data`
		`cantat_data = cantat_data.rename(columns={'Source':'source_id',`
		`'Proba':'PMemb_cantat'})`
		`# taking the subset of only source_id and PMemb`
		`cantat_data = cantat_data.loc[:,['source_id', 'PMemb_cantat']]`
		`# join the two table on source_id`
		`cantat_data = GAIA_target_stars.join(cantat_data.set_index('source_id'),`
		`on='source_id')`
		`# dropping the rows, where we don't have PMemb_cantat`
		`# (i.e. the source id was not in the cantat table)`
		`cantat_data = cantat_data.dropna(subset=['PMemb_cantat'])`
		`cantat_data.head()`

#NS125-DataAnalysis	Code Notebooks	Once we preprocessed the data by applying filter and selecting specific features, we focused on analyzing the data in the light of our hypothesis. Our main goal of this project is to develop a well-justified and optimized semi-supervised model. As an example, we apply GMM and RF to two different cluster (M67 and NGC3766) by considering the best practices. First before applying GMM to our model, we wanted to find the best value for distance cutoff and member threshold for the given dataset. I run the model with a range of different value, every time I run it several time (15 trials) to smooth out any random error and chose the best model. I also visualized the overall performance of the model for varying cutoff to see how the cutoff influences the performance (figure 4.5, 4.6, 4.6). These visualization helps to make a writing report where we can explain and justify our choice of cutoff.

		After applying the optimized GMM model, I calculated the MSS metric for the final model, that can be used for comparing with other models. I also visualized the group of members and field stars using their CMD, proper motion plot and parallax. These visualization helps to justify if GMM is performing well or not (i.e., the member group's proper motion should show a compact normal distribution, whereas the members should be scattered). Once we get the training data from GMM, before applying RF, we again analyzed the data to select the features (by applying SHAP values in python and applying a 95% threshold) and to optimize the hyperparameter (by performing cross-validation using our data). Again, for RF prediction, we produced useful visualization to support our justification and analysis. Altogether, following the best practices, we apply GMM and RF model for M67 and NGC3766. Then we compare our member prediction with the Cantat benchmark data. This visualization provides a very important insight that our model might be better to predict those members which are fainter and lies a bit further than the cluster center.
#NS125-LitReview	Section 2.5	The research question that motivated this paper is how we can confidently determine the members of an open star cluster from the direct observational data avoiding the field star contamination. In order to understand the current progress in this field and to find the existing gaps, we curated an extensive sample of recent literatures and synthesized the relevant information. As our specific research question concerns mainly regarding the ML models used to

		find these members, we focused on the methodology section of the papers and grouped the papers according to their use of ML model. First, we grouped them between the parametric and non-parametric ML models, and we found similar number of papers in both categories. Then we divided them into supervised and unsupervised models which showed a scarcity of supervised model. The reasoning of this scarcity is also discussed (not having a label training data). Lastly, we analyzed the effectiveness of the metrics they used to evaluate the model performance. By synthesizing the papers, two major limitations are identified that motivated our hypothesis. At the end, a well-organized summary has been provided by coherently summarizing the important findings.
#CS156-model-metrics	Mainly in Section 3.3, Also in section 2.5, 3.1, 7, Code Notebooks	First, I analyzed the metrics used in the past literature for the unsupervised model and found three most used approach: visual inspection with CMD, comparing with the findings of the past literature and finding misclassification rate in a simulated data. Then I discussed why these metrics are hard to use for direct comparing two unsupervised model just based on their predicted member and field star group. Visual metrics are prone to subjectivity. And the primary assumption behind comparing with past studies or simulation is that the past study or the simulation correctly represent the ground truth, which is not plausible most of the time. To test our hypothesis, we need to first find an optimal semi-supervised model, for which we need to compare different unsupervised models among themselves and similarly several supervised models.

		In section 3.3, I explained that based on the context, our priority is to reduce false positive (no field star should be assigned as a member), thus precision score is a good choice of evaluation metric for the supervised model. Now for the unsupervised model, as we do not have any common quantitative metrics, I proposed a new metrics: modified silhouette score, MSS. This metric assumes normally distribute members and uniformly distributed field stars and it can be used just using the predicted member and field star group. To provide more justification on the effectiveness of the model, we created simulated data with different number of stars, apply GMM membership and check how the MSS metric value changes depending on the poor or good clustering of the dataset. We found that when GMM make a bad separation (i.e., fail to separate the true member and field star), MSS value is very low, but when GMM performs better, MSS value becomes higher. Lastly, in conclusion, we acknowledged that the performance of the semi-supervised model will be mostly bottlenecked by the performance of the unsupervised model. Thus, when compared to other models, one way we can report the overall performance of the semi-supervised model is to by multiplying MSS with the precision value. Also, we should always try to use more than one metric, if possible, for more sophisticated report; thus in addition to precision, we can also use the area under the ROC curve to avoid both false positive and negative
#CS156-unsupervised-	Section 4, Code	In section 3.1, I mentioned why must use an unsupervised model in this context (as we do not usually have labels in the real data) and how it can be

learning	Notebook 2,3,4	combined with a supervised method to improve our membership detection (as evident from a few past papers). Before that, in the literature review, I discussed that it is very important to justify our choice of model and one way to do is to compare among several candidate models using an appropriate evaluation metric. To find our optimal semi-supervised model, we want to follow this approach. We will compare several unsupervised models using MSS metric. But before that we need to justify and optimize each of the supervised model. We use GMM as an example to show how we can do that. First, we need to understand the underlying assumptions. We show that GMM assumes all groups are normal distribution, thus we should not use photometric variables. While applying GMM, from the past literature and from our simulated data, we found that it is important to provide a half-width distance cutoff. We used MSS metric to justify the choice of the half-width value. We run multiple iterations and with different half-width value and chose the one with best MSS. Similarly once we have the membership probability, we can again use MSS to justify how strict or lenient cutoff should apply for selecting the training data. Lastly, in conclusion I discussed how we can apply other unsupervised models using the learning of this project and compare among them using MSS to find the best one.
#CS156-classification	Section 3.2, 5, Code	In section 3.2, we first discussed that why it is better to use a supervised classification algorithm instead of a supervised regression algorithm (as most unsupervised model outputs just groups of member and field star and do not

	Notebooks	provide any numerical membership probability). Similar to an unsupervised model, an optimal supervised model will be chosen by comparing several models together using the precision score. In this project, we used RF to show the best practices of using a supervised classification model. First, we need to select the features. We can use the feature importance with a 95% threshold value for feature selection. We discussed three different feature importance score (MDI, PFI, SHAP) and justify why it is better to use SHAP value for feature selection. As our training data could be biased (different number of member and field star) we used a stratified split to get the test and training data. Lastly, the model parameter should be optimized by the cross-validation. Also, as RF cannot do extrapolation, we applied the same filter used in the training data (GMM model) to our target data. But as ra and dec was not used as a feature in GMM model, we could apply RF to a larger region and the number of members significantly improved
#CS156-overfitting	Section 5.1, 5.3.2, 5.3.3	While applying any ML models, especially the supervised models, we need to always stay cautious of overfitting. Overfitting happens when a model explains the training data very well, but it fails to perform well in a new world data set. If we do not stay cautious, often we would not even realize that there is a overfitting. I explained in 5.1 that Decision Tree is a model which is very prone to overfitting as it tries to learn predicting the training data too well. I discussed how the use of an ensemble of decision tree and adding randomness in the dataset and the feature selection can reduce overfitting a lot. This is why RF is a good choice over DT to avoid overfitting. But even with RF, we

		need to consider two things to prevent overfitting. One, we always need to keep aside a fraction of the training data as a test subset, which we can use to get an unbiased performance evaluation. By comparing its performance (which we are measuring by precision here) in both training and test subset, we can understand if there is any overfitting or not. If there is overfitting, we will have more precision in training data than the test subset. But for an imbalance dataset, we need to make sure that the distribution stays same in test and train subset. This is why we used stratified split. Secondly, we need to optimize our hyperparameters to reduce the overfitting. In cross-validation, in every trial, we kept a validation test aside and train our model using the remaining data and finally measure the performance using the validation set. In every trial, we used different set of parameters and finally we chose the one which provided best validation score. We also used 5-fold cross-validation to optimize the hyperparameter for RF model in both cases with M67 and NGC3766 cluster. This reduced the overfitting and the final precision in both test and train set was similar for both cluster

HC Appendix

HC Rubrics can be found here: HC Rubrics

HCs	Place of application	Justification
#hypothesis-developement	Section 2.5 (evidence), 3.1 (prediction, mechanism, etc.), 3.2 (workflow) and 7 (falsifiability and future progress)	First, I reviewed the relevant literatures to find the current progress and gaps in the broad research question of how to find members of a star cluster. I found that the use of the supervised models is rare, and I pinpoint two major limitations that still need to be considered: the lack of comparative study and the lack of a common metric. From these literatures, I showed evidence that when both unsupervised and supervised are applied together, the detected number of members increased. Also, the ML literature suggests the power of supervised models to make a better prediction with proper validation metrics. Combining these evidence, I formulate my hypothesis with the following prediction. *If* we apply a well-justified and optimized semi-supervised algorithm, *then* the number of members and the model performance will improve compared to an unsupervised model. Then I explained the mechanism from the lens of how ML model works: *because* the confidence of unsupervised model increases with a less contamination of field stars, which results in a smaller number of members. But in a semi-supervised model we can use

		this small confident set as a training set to get more members in a larger region. Later I also explained and specified necessary components of the hypothesis including what is *well-justified and optimized model*, how I am planning to achieve this, what are specific outcome I want to compare. I also discussed the plausibility of the hypothesis as well as how I am planning to test this including the workflow (connecting with a study design). At conclusion, I come back again to my hypothesis to discuss what part of the hypothesis have been achieved, what are the future work needed to test the hypothesis. Here I also explained what results will support or oppose the argument. This discussion shows the falsifiability of my hypothesis. Lastly, I discussed the possible future studies for both cases when the hypothesis will be true or false.
#plausibility	Section 2.5 and 3.1	Two significant components of a hypothesis are its evidence and its mechanism. For justifying the evidence of my hypothesis, I first did an extensive literature review and specified the peer reviewed papers that directly backs the evidence. I specified three papers from membership determination context and two ML papers, which suggest the plausibility of the hypothesis. Then after laying out its prediction and mechanism, I specified one underlying logical argument in the mechanism. The symbolic representation of the argument is as follows:

		1. If A, then B 2. If B, then C 3. If C, then D 4. A 5. Therefore, D The argument is valid as it follows formal logic rules without any fallacies (If A, then B. A. Therefore B → If B, then C. B. Therefore C → If C, then D. C. Therefore D). After checking its validity, I also discussed the truth value of the premises to justify its soundness. Later I also specified another underlying assumption (supervised model can be confident with even a larger field star contamination) and explained its justification. From those discussion, I concluded that the hypothesis is highly plausible.
#testability	Section 3.1, 3.2, 3.3 and 7	I break down the components of the hypothesis to specify them and assess if they lead to a testable prediction. Firstly, I define the *well-justified and optimized semi-supervised model* and lay out a detailed plan for finding such a model combining optimal supervised and unsupervised models (especially in the workflow section). I also justify which past paper we can use as an example of an unsupervised model (Cantat-Gaudin et al., 2020) to compare with our model. Then I pinpoint the exact two outcomes that I want to measure. Using examples from the past literature, I

		explain why measuring the performance metric is difficult (especially for unsupervised models). Following that discussion, I proposed a new evaluation metric in section 3.3 to compare the unsupervised models. I also explain why it is testable both in principle and in practice. In conclusion, I lay out specific future works needed to test the hypothesis. I discuss the possible outcomes we can see and which of the outcomes will either support or oppose (falsify) our hypothesis.
#gapanalysis	Section 2.5, 3.1, 3.2, 3.3	In section 2.5, a thorough literature review has been done to know the progress and find any relevant gaps in the study of open star cluster membership determination. The models and metrics used in these papers have been analyzed to justify the identification of two limitations. First one is the lack of a comparative study. Current state is that there is no paper which compares between several candidate models to choose the optimal one. Goal state is to develop a strategy (or research design) to compare several candidate ML models in this context. One existing solution that I found in other contexts is to use a common metric to evaluate the performance of all candidate models in a problem and chose the best one. Due to its generalizability, this solution could be an effective choice to fill this gap as long as we justify the model selections and the choice of evaluation metric(s). Then I explained how I will only compare

		the unsupervised models among themselves using a chosen metric (MSS) and after making the models well-justified and optimized. Similarly, the supervised models will be compared among themselves after being optimized using precision. Second identified gap is the lack of a common metric for unsupervised model. Current state is that we do not have a common quantifying metric that can compare the performance of unsupervised models using only their predicted member and field stars. Goal state is to develop such a metric. First, we discussed why the current metrics (visual inspection, comparison, and simulation) are not suitable to compare different unsupervised models efficiently. Later in section 3.3, we thoroughly discussed one possible common metric to fill this gap.
#evidencebased	Mainly in Section 2.5, 3.1. Also in section 4.2, 5.3.1, 5.3.3. Empirical evidence are used in 3.3 and 4.3	Before developing the main hypothesis, an extensive review of recent and relevant peer reviewed papers has been done to find strong evidence for any of our claim. At the end of section 2.5, the major findings from the literature have been summarized clearly. Once we identified the gap and developed a plausible hypothesis, the main evidence (3 peer review paper from star cluster context and 2 from ML) behind the hypothesis are reiterated succinctly just before the hypothesis statement (first paragraph of section 3.1). Then in section 4.2, while explaining the applicability of GMM in this problem, I identified and presented

		the major conclusions from the peer reviewed past studies to support what modifications are necessary for a good performance of GMM. Similarly, while working with random forest, to support the use of randomized search for model optimization was supported by the findings of Bergstra & Bengio (2012). Another type of evidence that I used is the empirical results using a simulated data. To support the effectiveness of the MSS metric, I could not use any paper as it is a new proposed metric. Thus, I test this metric with several simulated data and presented my findings in a figure. I grouped the simulations based on three different MSS ranges (low, medium and high) to show how MSS can inform us about bad, moderate and good clustering in this context. Similarly, in section 4.3, I used results from the simulated data to support the argument that GMM works better for an optimal range of half-width value. If we provide very high or low half-width value, it will work poorly. An overall trend of how MSS varies with half-width values for different number of field stars were shown to show the effect of half-width in a succinct manner. Additionally, the GMM performance of a sample of simulation with different half-width value were presented to visually show the differences in performance. As the evidence from the simulation are not as robust as a peer reviewed paper, following this evidence, our final conclusions were qualitative

		instead of quantitative (i.e., instead of justifying the exact optimal half-width value, which cannot be supported by thus evidence, we only justified that there is an optimal region of half-width where GMM performs better).
#algorithm	Section 3.2, Code Notebooks	Firstly, following the hypothesis, we need to come up with a strategy to find the most optimal semi-supervised model. In section 3.2 (workflow), I discussed a specific algorithmic strategy to find the best semi-supervised model. There were two important parts in the strategy. One, how to connect both models together. We explained that as most unsupervised model provides only clustering (no membership probability) pathway-1 cannot be generalized. This is why we chose pathway-2, which will work for any type of unsupervised model. How we will get a final numerical output from any supervised model is also discussed. Second point is to how to compare among the models. As visual metric cannot be use without manual inspection, we preferred to use a common metric that can be applied to any unsupervised model, then explained and justified the model. In the code, for both GMM and RF, we justified the input (feature selection) as well as any model choice (hyperparameter optimization or finding distance cutoff). We ensured that all the decisions can be made without a need of visual inspection: distance cutoff and member cutoff can be selected by finding the

		model with best MSS value, feature selection of supervised model can be done using the SHAP value and 95% threshold, hyperparameter can be optimized automatically for any given dataset using the scikit-learn's implementation of cross-validation. In *Run_GMM_model* class, *get_member* method is implemented in a way so that it will work for any given set of features. Also, GMM only outputs two group, it doesn't say which one is member and which is field. So, in the algorithm, we used the assumptions that the group with the smaller average standard deviation is the member group. Based on the assumptions, we will automatically detect which one is member group (*Run_GMM_model* class: line 96-107). Lastly, useful and detailed docstrings are provided in all class and functions so that any external reader can read, understand and implement it.
#variables	Section 3.2, 5.3.3	In workflow sections, I identified the input variables (features or independent variable) for our problem (astrometric and photometric variables from GAIA) and our interested response variable (dependent variable or output, the numeric membership probability of the stars). I also discussed why we chose a continues variable as the output and what constraints are implied by this choice (i.e., cannot use a binary classifier). Because having a membership probability allow researcher to give more strict or lenient cutoff according to their research choice. Then considering

		this constraint, I explained how we can modify our choice of supervised model (bootstrapping to get the numeric PMemb from the binary output of classification). Also, while implementing the ML models, I discussed the hyperparameters of the model (n_component, n_init in GMM; max_depth, n_estimator, etc. in RF), which is different than the variables (GAIA dataset with astrometric and photometric) and discussed how we can select (i.e., as we have two groups, n_component will be 2 for GMM) or optimize (using cross validation) these parameters.
#dataviz	Throughout	For all of the visualizations, I provided brief but informative caption describing what we can see and infer from those visualizations. The caption, axis labels, legend, and color-coding followed the academic standard. While representing or interpreting the results of the ML models or justifying any of my choices with simulated data, I used proper visualizations of meaningful metrics. For example, when I explained the choice of MSS metric, I visually represented how an ideal distribution of member and field stars should be (3.1). Again, when I measure MSS of the different simulation, I grouped them based on the MSS metric as my purpose was to see if high MSS can be connected with the good performance of the model. While replication, I used the similar figures as Gao, with same axis limits, to make the comparison more suitable. Lastly in figure

		6.1, I presented the non-member in grey scale and overplot the members over them with blue color. This contrast of color makes it useful to understand what part of the non-member regions are not presented in the member distribution.
#design-thinking	Code Notebooks, Section 3.2	While implementing the GMM and RF methods, I had to constantly go through an iterative process of understanding the model (design), writing the codes (prototype), generating the results to find what works well and what seems wrong (evaluation). I also asked feedback from professors and peers and point experts to understand what can be improved (feedback incorporation). For example, I used SHAP value for feature importance by incorporating the feedback of Prof Ribeiro. Similarly, feedback from the committee meeting (that most unsupervised model do not give numerical output) helped me to choose pathway 2 instead of pathway 1 that I talked about in workflow. In the debugging of my python implementation, I go through several iterations to solve problems that raised after primary round. For example, when I first made the *get_MSS_metric,* I found that even when there is no member group, it outputs 1. After several changes in the code and testing, I realized that when member group has no stars, its standard

		deviation becomes None and NumPy ignores SD_{member} from calculation, resulting in an output of 1. Then I added another condition that if any group has less than 2 stars (which means a bad performance as ideally there always will be more than 2 members or field stars but the model groups almost all of them in a single group), then MSS will be lowest, i.e. 0.
#study-replication	Section 3.3	For each of the models, I replicated one existing paper that used that model. The main reason behind the replication was first to understand the implementation of the model in this context (membership detection), then I compared the findings and plots of my replication with the original paper to show that I correctly replicated Gao paper. Then I suggested modifications to each of the model to improve justification of the original member. For GMM the use of MSS and distance half-width improve our justification. Similarly for RF, SHAP value for feature selection and crossvalidation for hyperparameter optimization helped improved the justifications of the Gao paper. Adding those modifications also allow us to automate the process of any decision without direct visual inspection or trial and error. For example, for selecting member cutoff in GMM outputs, we directly used MSS and took the cutoff with the largest MSS.
#optimization	Section 5.3.3	In section 5.3.3, we wanted to optimize the hyperparameters for RF for mainly two reasons. The specific

		choice of those hyperparameters influences our model performance, thus, we wanted to choose a model which provides the best performance. Another critical point is that we wanted to avoid overfitting our model, so we do not want to have a model which perfectly trained to classify training data but poorly classify any new data. In this context, our optimization problem is as follows. The function that we want to optimize (in this case, maximize) is the model performance. We used precision as our chosen metric for model performance. So, our objective is to maximize the precision score for our model. The variables that we can change to find the most optimal models are the specific model hyperparameters. We explained in 5.1. how different model parameters influence the model and its performance. The chosen set of variables (hyperparameters) and their possible ranges are listed in Table 5.2. Next, we had two major considerations in this problem, which can be assumed as some constraint in this context. One, as we wanted to reduce overfitting, I cannot just use the same data that I used to train my model to evaluate the performance of the model. If we do that, our model will be biased to this training dataset and become too specific for the given training dataset. Then it would show a high performance in training data but poor performance in any test data. We can try to solve this problem is to

		divide the training subset into two subsets: the *train* subset and the *validation* subset. Then we can use the *train* subset to train our model and the *validation* subset to evaluate the model performance. But this is where our second consideration comes in. Secondly, we want to use as much data as possible to train our model because more training data means more information to learn from; thus, it most often increases the model performance. If we use a different train and validation subset, we are not using another fraction of the data for our training purpose. This is also something we do not want to do. Following these two considerations, one best way to optimize our model is to use k-fold cross-validation, which we used in this paper. As explained in 5.3.3, in k-fold cross-validation, we divide the training data into k splits, and every time use the k-1 splits to train our model and use the last split for evaluation. We do this for k times, so every split is used as an evaluation set once, then take the average of the evaluation metric (precision in our case). In this way, we are not using the same data for training and evaluation while making use of the full training data. We used a 5-fold cross-validation with 100 (for M67) and 50 (NGC3766) iterations. Lastly, in cross-validation, we can either make a grid of parameter values and test with all combinations of parameters

		from the grid. Or we can select a range (or list) for each parameter and in every iteration, we can take a random sample from the range (or the list) for each parameter. We used random-search instead of grid-search because it has two advantages as shown by (Bergstra & Bengio, 2012). By applying this randomized 5-fold cross-validation, we could reach local maxima for precision. We could not guarantee the global maxima, as we are doing a randomized search with finite iterations. If we increase the iterations, the possibility of reaching a global maxima increases.
#probability	Section 4.1, 5.1	In the explanation of the GMM model (Section 4.1), I showed, using the Bayes' theorem, how GMM can predict the membership probability for each of the observations from the distribution parameters. Given that we know the feature values, x_i for a certain observation, i, then the probability that this observation is into group k, can be expressed by a conditional probability: $p(c_k \mid x_i)$. This conditional probability can be expanded using Bayes' theorem in terms of joint and marginal probability as explained in section 4.1. In simple terms, we want to find the posterior probability of being into (or generated from) the group k, given the new data (that is the feature value of the observation). We can find that if we know the prior probability without any data and the likelihood.

		In this context, the prior probability is the following. If we randomly chose an observation, what would be its probability of being in group k. This is defined as the mixture weight, w_k in the GMM. In GMM, every group k is assumed to generated by a multi-dimensional Gaussian distribution with a mean vector and a variance matrix. So, if we know (or have an estimate of) the mean and variance, we can get the likelihood. Lastly, we need to normalize
#sourcequality	Mainly Section 2.5	In this project, I tried to tackle a research question by laying out a specific, plausible, and falsifiable hypothesis (an optimal semi-supervised model improves the member determination of an open cluster), defining the specific way to test it (by specifying the workflow and the justifying the evaluation metric) and making some analysis to progress on testing the hypothesis (by implementing at least one unsupervised (GMM) and supervised (RF) model). As an academic work, it has to build upon solid and strong evidence. Thus, the sources from which the supporting claims are taken have to be peer-reviewed academic journals. In the literature review (section 2.5), to understand the current progress, I synthesized a large number of recent and most relevant peer-reviewed articles and analyzed their use of ML models and the evaluation metrics. Based on the collective review

		of the journals, I detected two specific limitations which motivated our hypothesis and the specific workflow (comparing different models and using a common unified metric to do that). To support the main evidence of my hypothesis, I provided five peer-reviewed articles. In an academic setting, the hypothesis would not be well supported if we used any popular articles (tutorials, blog posts, or medium articles), because they are not reviewed and approved by the experts in the field. When we just provided information and definitions on a background topic, having a peer-reviewed paper is not essential. A reference from a classic textbook will be enough for this cause as we are not using it to make a new hypothesis. For example, when explaining what an open star cluster is, we used a classic textbook from the Cambridge press (Platais, 2009). Similarly, when explaining the advantage and limitations of parametric and non-parametric models, I used a textbook reference published by Springer (James et al., 2017). But during the research work and analysis, if we want to justify any particular choice or decision by a source, it is recommended to use recent journal articles. Because if I am doing primary research on a particular topic, it means that I am trying to provide some new idea in the current active field. Thus, for any major decision, I should rely on only the claims that are approved

		by the field experts. While running GMM, to support the claim of how we should modify GMM for membership determination (section 4.2), I used two articles (Cabrera-Cano & Alfaro, 1990; de Graeve, 1979). In cross-validation to optimize the hyperparameters (section 5.3.3), I used the claims of Bergstra & Bengio (2012) to justify the choice of a randomized search instead of a grid search.
#audience	Throughout the paper	I kept in mind that my target audience is my peers who have taken the same sort of courses as me. So, I avoided complex technical or astronomical jargon. When necessary, I provided a definition first. Also, as my target audience is peers, I shared my writings with them to get feedback on if it is understandable enough. For example, I provided a very simple and short definition of a star cluster. As it was not necessary for this paper, I did not go in-detail explanation of how star clusters are formed from gas clouds and how they evolve in their lifetime. Similarly, for ML models, I explained in a simple term what are parametric vs non-parametric models and supervised vs unsupervised models, then I just mentioned which specific ML models go to which category. As it is not necessary, I did not explain the underlying assumptions of every single ML model. At the end of the literature review (Section 2.5), I provided a simple summary without any technical

		term that can be easily understood. When it was necessary to go into details (Section 4.1, 5.1, 5.3.1), I do not assume any prior technical knowledge for my audience other than the one taught in the Minerva courses. So, I started from a very primary level and explained with simple terms. For example, while explaining the GMM model in 4.1, I used the one-dimensional normal distribution (that we learned in CS146) to build into multi-dimensional gaussian distribution. Similarly, while explaining SHAP values in 5.3.1, I started with the most basic combinatorics equation (i.e., $C(n,p) = \frac{n!}{(n-s)!\, s!}$), because I did not have combinatorics in my Minerva courses.
#organization	Throughout the paper	The paper has been organized in a way that is easy to understand. After introducing the problem context (determine the member of open star clusters) in the introduction, I moved on to explain the relevant background discussion. I wanted the paper itself to be self-contained, thus I provided the definitions of what an open cluster is, what are the different types of variables (astrometric and photometric) we used to solve the problem, and their advantage and limitations. As I want to use an ML model to solve the problem, I provided the necessary background on different types of ML models along with the literature review. Then in order to properly highlight the main hypothesis and research design, I had a full section where I explained my

		specific focus. I clearly defined my hypothesis and discussed its plausibility and the specific way we can test it. Then the workflow is explained with the help of a flow chart. Similarly, the choice of evaluation metric has been proposed and justified in a different subsection to reiterate its importance. After that I have two major sections, where I explained my analysis of GMM and RF. For both sections, I first discussed the underlying assumptions of the model and how they influence some decisions for the analysis. For example, as the RF model cannot do interpolation, we need to apply the same filter that we used in the training data to the target data. I also defined the important model parameter to understand how they influence the model performance (i.e., decreasing maximum depth helps us to reduce overfitting in the cost of an increase of variance in RF). Then I have different subsections for my detailed analysis and the results for the direct replication, modified replication, and the use of the model in a new cluster. I had a small section on results to report the most important findings of the analysis and their comparison. I made it a separate section so that for the reader it becomes very easier to check and compare the final findings even without going into the much-detailed discussions of GMM and RF in section 4 and 5. Lastly, in the conclusion section, I connected my analysis

		with the original hypothesis again. I showed how much progress we have been done to test the hypothesis., Now we have a systematic strategy that we can use to justify and optimize different unsupervised and supervised models before finding the most optimal one from each category. I also discussed the specific future works needed for testing of the hypothesis. I concluded by explaining what are the possible outcomes that will support or oppose the hypothesis and in either case what could be next research question to explore.

Capstone LOs

Capstone LO Rubrics can be found here: Capstone LO

LOs	Place of Application	Justification
#quality-deliverables	Throughout the paper	This project tackles a primary research question starting from an extensive literature review to developing a well-justified hypothesis and then laying out the plan for how to test the hypothesis. Testing the hypothesis while strongly justifying every decision is crucial for any scientific research. Constrained by the limited time, I explored two ML models with in-detail analysis of their underlying assumptions and justifying my choices. Then I implemented these models in two different clusters. All four code notebooks are organized and documented sufficiently. Lastly, I provided the specific future works needed to complete testing this hypothesis and how we can proceed further.
#curation	Throughout the paper	As my project is a primary paper, I arranged my final report according to the structure of a primary paper in astronomy field. I provided necessary background information to understand the project, then added a literature review and succinct summary. Hypothesis was the most important part of this project, so I explained and justified it in a separate section. Later I divided my sections based on the two ML models that I implemented. In conclusion, I again tied back everything to the hypothesis to lay

		out the future direction. I added a table of content and numbered the section and subsection so that reader can navigate it easily. This also helped me to reference any particular subsection. I did not paste my full code notebook in the project as it will hamper the flow of the paper. But in appendix, I provided the structured and commented code snippets in a way that is easy to understand. Lastly in HC/LO appendix, I divided them into three subsections so that one can come back to a specific subsection directly from the table of contents.
#metrics	HC/LO appendix	I identified and justified the relevant HCs and LOs that I applied in this project both in terms of implementing the python code and writing the report. Whenever possible I connected the justifications of the LO/HCs to specific examples of the project or code notebooks. During the last year, I constantly tried to incorporate feedback from the professors and peers to improve my application and justification of the LO/HCs. I also identified the completeness of the project itself at its final stage and explained what are the future works needed to test the given hypothesis and how we can achieve this (i.e., how we will compare several models to find the optimal semi-supervised model, etc.) Throughout the year, in the advisory and committee meeting presentations and the assignments, I mentioned the remaining

		works with a practical future plan to complete those work in time.
#navigation	Throughout the process	In order to keep myself accountable, I set a weekly co-working time to meet with Jatin. We added the time blocks to our calendar, send invitations and also list our progress in a spreadsheet. We found that two of the hardest part of the co-working session were the followings. One, during most of the co-working time, we just think about what part we should focus on. So, we decided that we will decide at the end of the meeting, which topic we should work on the next meeting. Second, when one is absent in the meeting, other people can also feel less motivated and not work during the time. So, we decided that if any one of us did not show up, we will call or go to his room (if we are in the same place). In this way, we can keep ourselves motivated and keep the others accountable.

Bibliography

Agarwal, M., Rao, K. K., Vaidya, K., & Bhattacharya, S. (2021). ML-MOC: Machine Learning (kNN and GMM) based Membership determination for Open Clusters. Monthly Notices of the Royal Astronomical Society, 502(2), 2582–2599. https://doi.org/10.1093/mnras/stab118

Balaguer-Núñez, L., Tian, K., & Zhao, J. (1999). Determination of proper motions and membership of the open clusters NGC 1817 and NGC 1807. Astronomy and Astrophysics Supplement Series, 133(3).

Balakrishnan, S., Wainwright, M. J., & Yu, B. (2017). Statistical guarantees for the EM algorithm: From population to sample-based analysis. *The Annals of Statistics*, 45(1), 77-120. https://doi.org/10.1214/16-AOS1435

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of machine learning research*, *13*(2). http://jmlr.org/papers/v13/bergstra12a.html

Bhattacharya, S., Mahulkar, V., Pandaokar, S., & Singh, P. K. (2017). Morphology of open clusters NGC 1857 and Czernik 20 using clustering algorithms. Astronomy and Computing, 18. https://doi.org/10.1016/j.ascom.2016.10.001

Bossini, D., Vallenari, A., Bragaglia, A., Cantat-Gaudin, T., Sordo, R., Balaguer-Núñez, L., Jordi, C., Moitinho, A., Soubiran, C., Casamiquela, L., Carrera, R., & Heiter, U. (2019). Age determination for 269 Gaia DR2 open clusters. Astronomy & Astrophysics, 623, A108. https://doi.org/10.1051/0004-6361/201834693

Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32. https://doi.org/10.1023/A:1010933404324

Cabrera-Caño, J., & Alfaro, E. J. (1990). A non-parametric approach to the membership problem in open clusters. *Astronomy and Astrophysics, 235*, 94-102. (PDF) A non-parametric approach to the membership problem in open clusters

Cantat-Gaudin, T., Jordi, C., Vallenari, A., Bragaglia, A., Balaguer-Núñez, L., Soubiran, C., Bossini, D., Moitinho, A., Castro-Ginard, A., Krone-Martins, A., Casamiquela, L., Sordo, R., & Carrera, R. (2018). A Gaia DR2 view of the open cluster population in the Milky Way. *Astronomy & Astrophysics, 618*, A93. https://doi.org/10.1051/0004-6361/201833476

Cantat-Gaudin, T., & Anders, F. (2020). Clusters and mirages: Cataloguing stellar aggregates in the Milky Way. *Astronomy and Astrophysics. 633*, A99. https://doi.org/10.1051/0004-6361/201936691

Castro-Ginard, A., Jordi, C., Luri, X., Cantat-Gaudin, T., & Balaguer-Núñez, L. (2019). Hunting for open clusters in Gaia DR2: The Galactic anticenter. *Astronomy and Astrophysics, 627*. https://doi.org/10.1051/0004-6361/201935531

Castro-Ginard, A., Jordi, C., Luri, X., Cid-Fuentes, J. Á., Casamiquela, L., Anders, F., Cantat-Gaudin, T., Monguió, M., Balaguer-Núñez, L., Solà, S., & Badia, R. M. (2020). Hunting for open clusters in Gaia DR2: 582 new open clusters in the Galactic disc. *Astronomy and Astrophysics*. https://doi.org/10.1051/0004-6361/201937386

Castro-Ginard, A., Jordi, C., Luri, X., Julbe, F., Morvan, M., Balaguer-Nuñez, L., & Cantat-Gaudin, T. (2018). A new method for unveiling open clusters in Gaia New nearby open clusters confirmed by DR2. *Astronomy and Astrophysics, 618*. https://doi.org/10.1051/0004-6361/201833390

de Graeve, E. (1979). Astrometric criteria for selecting "physical members" of open clusters with low astrometric precision: application to NGC 559. *Vatican Observatory Publication, (1),* 283-306. https://ui.adsabs.harvard.edu/abs/1979VatOP...1..283D/abstract.

El Aziz, M. A., Selim, I. M., & Essam, A. (2016). Open cluster membership probability based on K-means clustering algorithm. *Experimental Astronomy, 42(1)*, 49–59. https://doi.org/10.1007/s10686-016-9499-9

Gaia Collaboration, Brown, A. G. A., Vallenari, A., Prusti, T., De Bruijne, J. H. J., Babusiaux, C., Bailer-Jones, C. A. L., Biermann, M., Evans, D. W., Eyer, L., Jansen, F., Jordi, C., Klioner, S. A., Lammers, U., Lindegren, L., Luri, X., Mignard, F., Panem, C., Pourbaix, D., Randich, S., … Zwitter, T. (2018). Gaia Data Release 2: Summary of the contents and survey properties. In arXiv. https://doi.org/10.1051/0004-6361/201833051

Gao, X. (2014). Membership determination of open cluster NGC 188 based on the DBSCAN clustering algorithm. *Research in Astronomy and Astrophysics, 14(2)*. https://doi.org/10.1088/1674-4527/14/2/004

Gao, X. (2018a). A Machine-learning-based Investigation of the Open Cluster M67. The Astrophysical Journal, 869(1). https://doi.org/10.3847/1538-4357/aae8dd

Gao, X. (2018b). Memberships, distance, and proper motion of the open cluster NGC 188 based on a machine learning method. *Astrophysics and Space Science, 363(11)*, 232. https://doi.org/10.1007/s10509-018-3453-4

Gao, X. (2018c). Memberships of the open cluster NGC 6405 based on a combined method: Gaussian mixture model and random forest. *The Astronomical Journal, 156(3)*, 121. https://iopscience.iop.org/article/10.3847/1538-3881/aad690/meta

Gao, X. (2020). 5D memberships and fundamental properties of the old open cluster NGC 6791 based on Gaia -DR2. *Astrophysics and Space Science, 365(2)*. https://doi.org/10.1007/s10509-020-3738-2

Hidayat, Y. A., Arifyanto, M. I., Aprilia, & Hakim, M. I. (2019). An Application of The Markov Chain Monte Carlo (MCMC) Method to Open Cluster Membership Determination. Journal of Physics: Conference Series, 1231(1), 012029. https://doi.org/10.1088/1742-6596/1231/1/012029

Jackson, R. J., Jeffries, R. D., Wright, N. J., Randich, S., Sacco, G., Pancino, E., Cantat-Gaudin, T., Gilmore, G., Vallenari, A., Bensby, T., Bayo, A., Costado, M. T., Franciosini, E., Gonneau, A., Hourihane, A., Lewis, J., Monaco, L., Morbidelli, L., & Worley, C. (2020). The Gaia-ESO Survey: Membership probabilities for stars in 32 open clusters from 3D kinematics. *Monthly Notices of the Royal Astronomical Society.* https://doi.org/10.1093/mnras/staa1749

Jain, A. K., Murty, M., & Flynn, P. (1999). Data clustering: A Review ACM Computing Surveys, vol. 31. Google Scholar, 264-318. http://users.eecs.northwestern.edu/~yingliu/datamining_papers/survey.pdf

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2017). *An introduction to statistical learning* (Chapter 2). New York [etc.]: Springer. ISBN-10: 1461471370.

Keshav, S. (2007). How to read a paper. ACM SIGCOMM Computer Communication Review, 37(3), 83-84. Retrieved from http://ccr.sigcomm.org/online/files/p83-keshavA.pdf

Kharchenko, N. V., Piskunov, A. E., Röser, S., Schilbach, E., & Scholz, R. D. (2005). Astrophysical parameters of Galactic open clusters. *Astronomy & Astrophysics, 438(3),* 1163-1173. https://doi.org/10.1051/0004-6361:20042523

Lee, K. J., Guillemot, L., Yue, Y. L., Kramer, M., & Champion, D. J. (2012). Application of the Gaussian mixture model in pulsar astronomy-pulsar classification and candidates ranking for the Fermi 2FGL catalog. *Monthly Notices of the Royal Astronomical Society*, 424(4), 2832-2840. https://doi.org/10.1111/j.1365-2966.2012.21413.x

Liu, L., & Pang, X. (2019). A catalog of newly identified star clusters in Gaia DR2. In arXiv. https://doi.org/10.3847/1538-4365/ab530a

Louppe, G. (2014). Understanding random forests: From theory to practice. *arXiv preprint arXiv:1407.7502*. https://arxiv.org/abs/1407.7502

Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. *Advances in neural information processing systems*, *30*. Retrieved from https://proceedings.neurips.cc/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf

Mahmudunnobe, M. (2020a). "Cluster Membership", Round 1 Project Exploration Report, CP191: Capstone Seminar, Minerva University. Retrieved from https://bit.ly/3c8GI6I

Mahmudunnobe, M. (2020b). "Membership Detection of Open Star Cluster NGC 3766 using Random Forest Model", Final Project Assignment, CS156: Machine Learning for Science and Profit, Minerva University. Retrieved from https://tinyurl.com/ykvnjzu4

Mahmudunnobe, M., Hasan, P., Raja, M., & Hasan, S. N. (2021). Membership of stars in open clusters using random forest with Gaia data. *European Physical Journal: Special Topics*, *230*(10). https://doi.org/10.1140/epjs/s11734-021-00205-x

Maíz Apellániz, J. (2019). Gaia DR2 distances to Collinder 419 and NGC 2264 and new astrometric orbits for HD 193 322 Aa, Ab and 15 Mon Aa, Ab. *Astronomy & Astrophysics, 630,* A119. https://doi.org/10.1051/0004-6361/201935885

Mermilliod J.-C. (1995), The database for galactic open clusters (BDA). *Information and On-Line Data in Astronomy (Kluwer Academic Press, Dordrecht), p. 127-138 https://webda.physics.muni.cz/webda.html*

Monteiro, H., & Dias, W. S. (2019). Distances and ages from isochrone fits of 150 open clusters using Gaia DR2 data. *Monthly Notices of the Royal Astronomical Society, 487(2),* 2385–2406. https://doi.org/10.1093/mnras/stz1455

Netopil, M., Paunzen, E., & Stütz, C. (2012). Developments of the open cluster database WEBDA. In *Star Clusters in the Era of Large Surveys* (pp. 53-61). Springer, Berlin, Heidelberg. https://webda.physics.muni.cz/extension.html

Ochsenbein, F. (1996). The VizieR database of astronomical catalogues. CDS, Centre de Données astronomiques de Strasbourg. https://doi.org/10.26093/CDS/VIZIER

Pedregosa, F., Varoquaux, Ga"el, Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*(Oct), 2825–2830. Retrieved from https://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html

Perren, G. I., Vázquez, R. A., & Piatti, A. E. (2015). ASteCA: Automated Stellar Cluster Analysis. *Astronomy & Astrophysics, 576, A6.* https://doi.org/10.1051/0004-6361/201424946

Perryman, M. A. C., De Boer, K. S., Gilmore, G., Høg, E., Lattanzi, M. G., Lindegren, L., Luri, X., Mignard, F., Pace, O., & De Zeeuw, P. T. (2001). GAIA: Composition, formation, and evolution of the Galaxy. *Astronomy and Astrophysics, 369(1).* https://doi.org/10.1051/0004-6361:20010085

Platais, I. (2009). Star clusters. In *Astrometry for astrophysics: Methods, Models, and Applications* (pp. 360–367). Cambridge University Press. https://doi.org/10.1017/CBO9781139023443.026

Reddy, Y. C. A. P., Viswanath, P., & Reddy, B. E. (2018). Semi-supervised learning: A brief review. Int J Eng Technol, 7(1.8), 81. https://doi.org/10.14419/ijet.v7i1.8.9977

Sampedro, L., & Alfaro, E. J. (2016). Stellar open clusters' membership probabilities: an N - dimensional geometrical approach. Monthly Notices of the Royal Astronomical Society, 457(4), 3949–3962. https://doi.org/10.1093/mnras/stw243

Sim, G., Lee, S. H., Ann, H. B., & Kim, S. (2019). 207 new open star clusters within 1 kpc from gaia data release 2. Journal of the Korean Astronomical Society, 52(5). https://doi.org/10.5303/JKAS.2019.52.5.145

Stott, J. (2018). Determining open cluster membership: A Bayesian framework for quantitative member classification. Astronomy and Astrophysics, 609. https://doi.org/10.1051/0004-6361/201628568

Wenger, M., Ochsenbein, F., Egret, D., Dubois, P., Bonnarel, F., Borde, S., ... & Monier, R. (2000). The SIMBAD astronomical database-The CDS reference database for astronomical objects. *Astronomy and Astrophysics Supplement Series*, *143*(1), 9-22 https://doi.org/10.1051/aas:2000332

Yadav, R. K. S., Sariya, D. P., & Sagar, R. (2013). Proper motions and membership probabilities of stars in the region of open cluster NGC 3766. Monthly Notices of the Royal Astronomical Society, 430(4), 3350–3358. https://doi.org/10.1093/mnras/stt136

Yen, S. X., Reffert, S., Schilbach, E., Röser, S., Kharchenko, N. V., & Piskunov, A. E. (2018). Reanalysis of nearby open clusters using Gaia DR1/TGAS and HSOY. Astronomy & Astrophysics, 615, A12. https://doi.org/10.1051/0004-6361/201731905