CP193 Full Draft

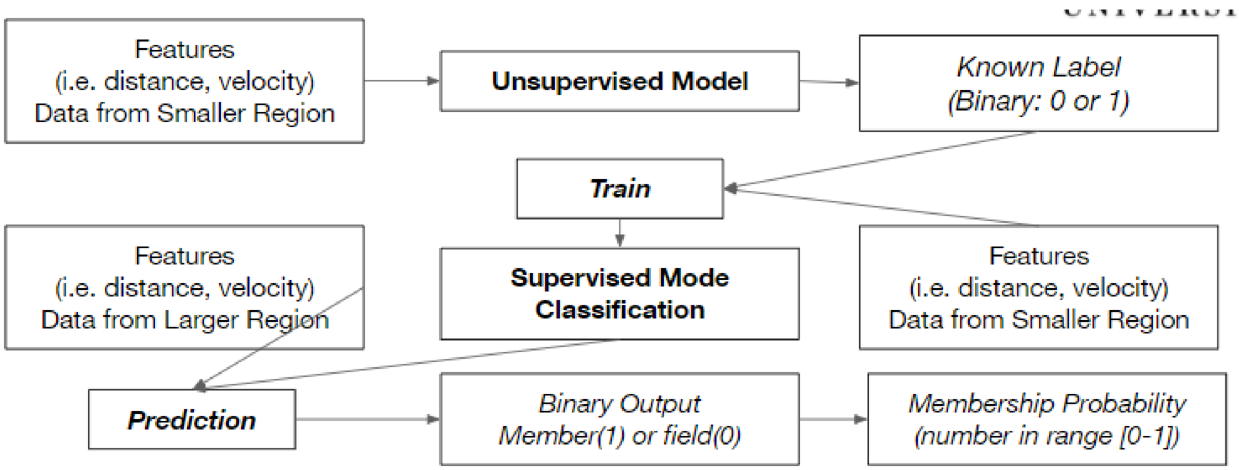
A Semi-supervised Machine Learning Approach for Open Star Cluster Member Determination

Md Mahmudunnobe

Minerva Schools at KGI

Executive Summary

When a group of stars born together from a same cloud of gases, it is called a star cluster. Analysis and research on star cluster helps us understand the formation of stars and how they evolve over time. But we need to first know which stars are the member of a cluster before analyzing them. Most of the available methods for member determination used a unsupervised approach, where we divide the stars in two or more separate groups based on their properties. The supervised methods are usually better in terms of efficiency and validation, but we need a labelled training data for that. As shown in the flowchart, I am proposing to use 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

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 a 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 between 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 with 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 star 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 section, 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 supervised model. Then we conclude with the future work in section 6.

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 a 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 in y-axis and difference between BP band and RP band magnitude (BP-RP color) in x-axis. We can see that the top of the CMD is changing slightly as the age of the stars are increasing. 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.

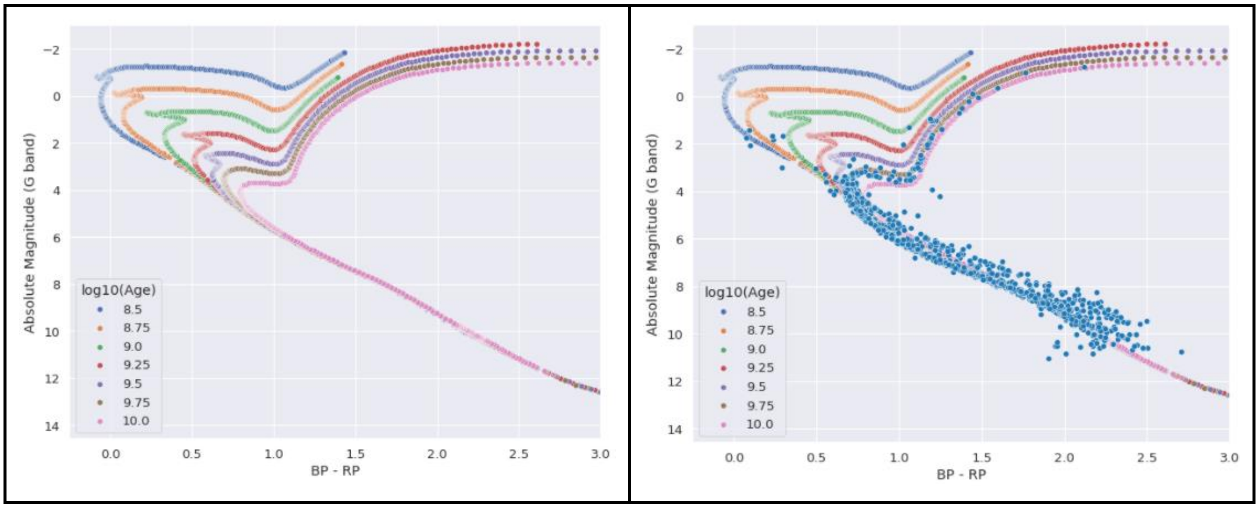


Figure 2.1: Theoretical HR or Color-Magnitude Diagram (CMD) for varying age. In the right, a set of cluster members are plotted as blue dots, which closely follows the CMD.

The diagonal line of the CMD that goes from the bottom-left to top-right is called the main sequence line. The region on the top with the horizontal lines is called the supergiant region. A 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. Once a star finished burning all effective hydrogen in its core and start burning helium, it enters 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 start going towards the giant region is defined as the ‘turning point’. Depending on the mass of a star, its duration as a 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. The stars that are currently on the turning point of CMD, their main sequence lifetime is equal to the age of the cluster. 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, the more the surface gases are attracted to the core. To balance a larger inward pressure, the star needs a larger outward pressure, which requires burning more hydrogens in a unit time. Luckily, the more inward pressure increases, the density and temperature of the core increases, which in turns 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 more brighter in the sky. As a more massive star has a higher nuclear reaction rate, it finishes burning all core hydrogen faster, thus have a short duration in main sequence. So, if we can measure the brightness of the star in 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.

Another 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 the 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 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:* difference between BP and RP magnitude (or BP - RP color)
5. *BP-G:* difference between BP and G magnitude (or BP - G color)
6. *G-RP:* 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 the astrometric data in about micro arcseconds accuracy (Perryman et al. 2001; Gaia Collaboration et al. 2018), which opens 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). I will also use the early 3rd data release of the Gaia mission in my project, 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 a 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 less 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 need training data with labeled output are called the supervised learning models and the ones that work without any labeled training data are unsupervised learning models. Most of the above-mentioned models are unsupervised learning models, i.e., 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). 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.

3. Project Description

3.1. Specific Focus: Hypothesis

Machine learning literature suggests that supervised models are better to validate and to get 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) used an unsupervised model first to get the training data and then fed that in a supervised model to get their final membership probabilities. Both papers reported an increase in members in this approach.

Based on this evidence, I argue that a combination of an unsupervised and a supervised model would be a better approach to detect members of open star clusters. Here, by better approach, I meant that this approach should increase the accuracy in distinguishing between the member and field stars for open clusters, and for many cases, it should also find more unfound members.

Now one problem is to understand which supervised method and which unsupervised method we should use. Most of the past papers chose only a single model for their but did not compare between other possible models. To fill this gap, I am planning to compare several unsupervised models to choose the best one. Considering the time constraint, I chose only two such models: GMM and UPMASK. Once I got the best-unsupervised model, I will similarly compare several supervised models to choose the best model. Again, for time constraints, I am only considering RF and Support Vector Machine (SVM).

Lastly, I will run this combined model of chosen unsupervised and chosen supervised model for several open clusters and compare the number of found members with previous literature to test my hypothesis.

3.2. Workflow

Figure 3.1 shows a clearer picture of how I am planning to use these both supervised and unsupervised together. Firstly, I am taking 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.

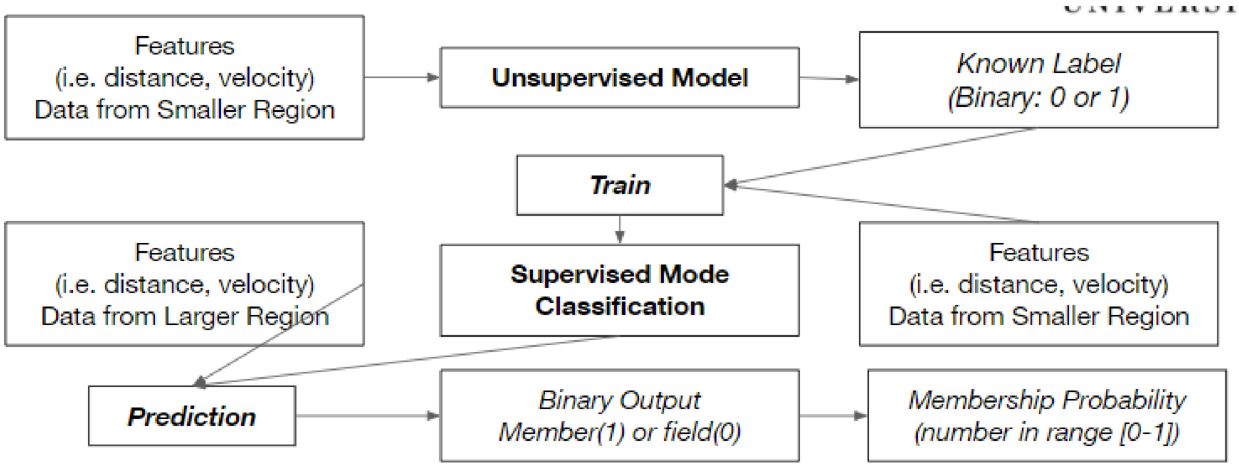


Figure 3.1: The flowchart of my project shows how I am planning to use both unsupervised and supervised methods (Pathway - 2).

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. So, 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 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 how the models are doing to distinguish between member and field stars. This evaluation metric can also be used to hyper tuning the model parameters which we will discuss later. Another point to note is that we will have a set of possible members and a set of field stars once we apply a cutoff. So, 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 score 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.

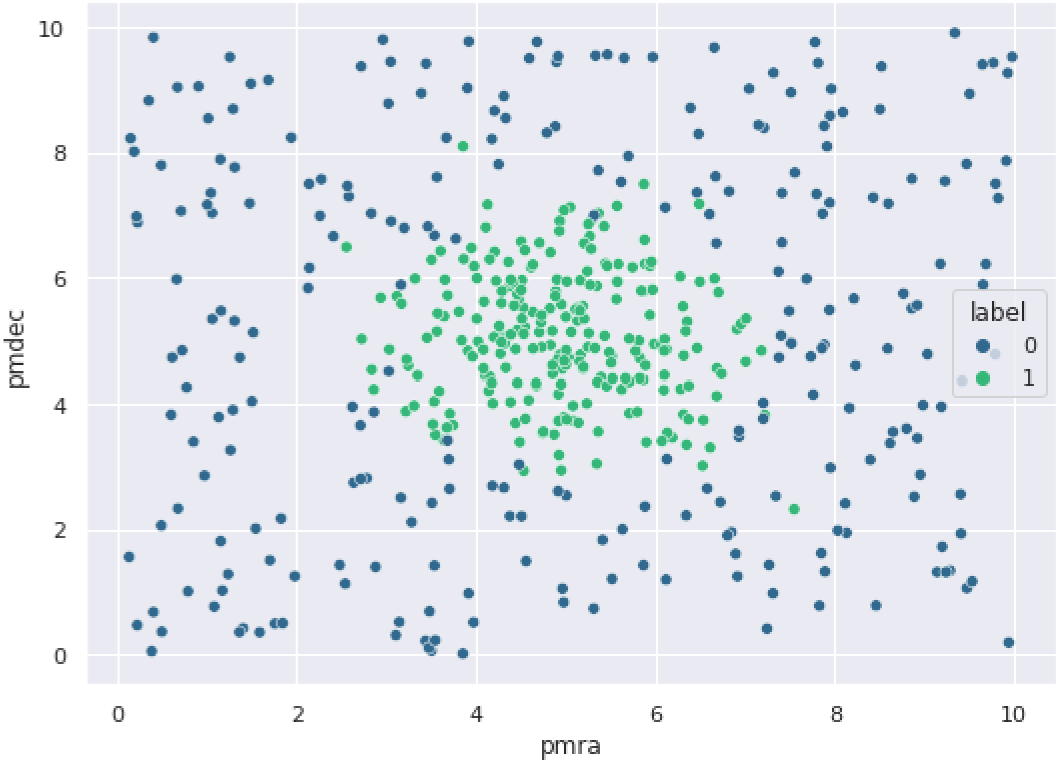


Figure 3.2: A simulated data showing how the member and field stars should ideally be distributed in their variable space. Here label = 1 are the members, which are compact and normally distributed, while the field stars are distributed all over places like a uniform and random distribution.

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 should be quite small. On the other hand, the field stars are uniformly distributed, so their standard deviation (SD) should be very large. Thus, standard deviation 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 standard deviation 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,

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

Fig. 3.3 shows how MSS performs for a set of simulated datasets.

Another useful metric would be to compare our findings with the past literature. The most recent paper of determining 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.

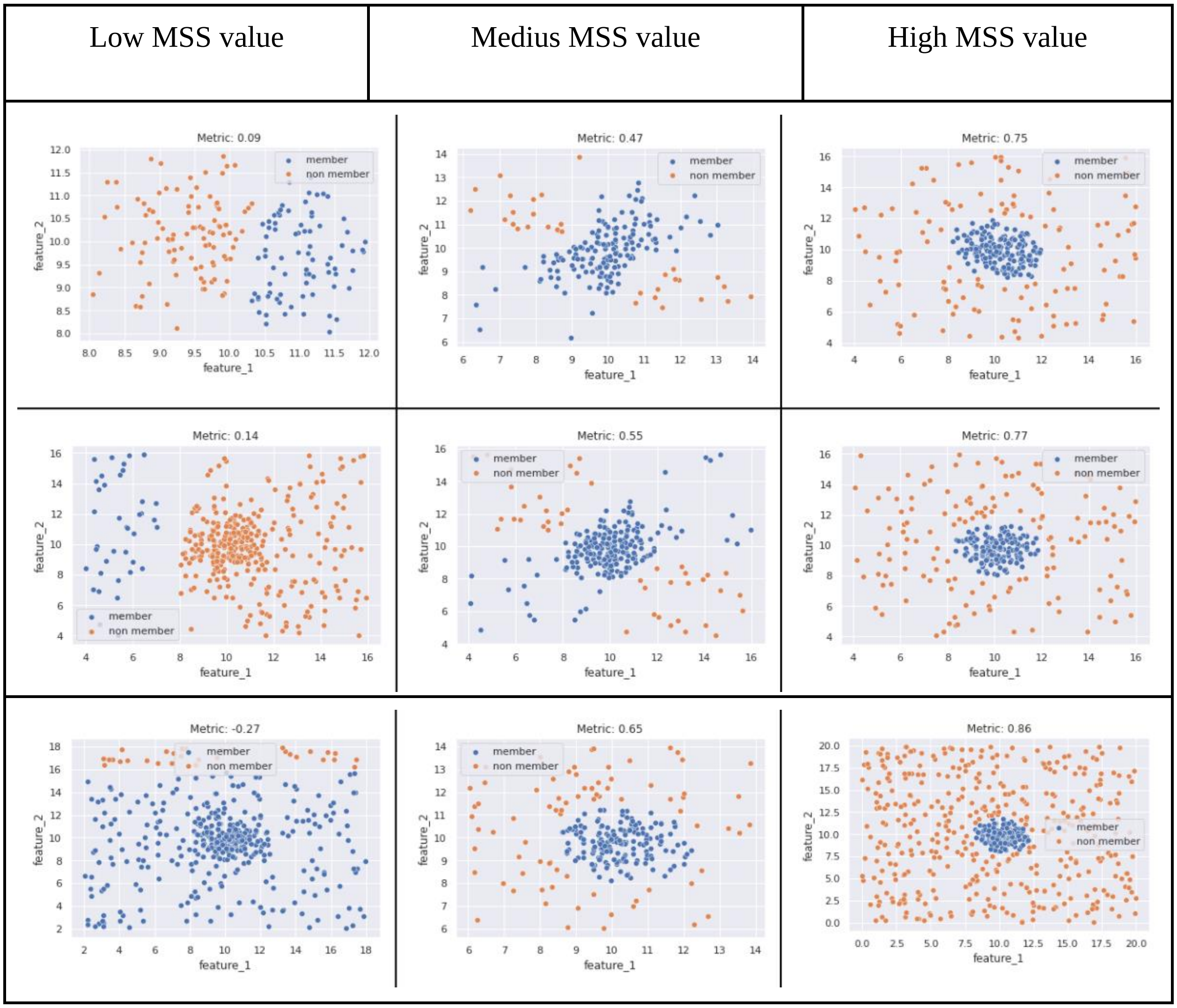


Fig 3.3: MSS Metric values for a range of different simulated setup.

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 $\boldsymbol{X} = \{\boldsymbol{x_1}, \mathbf{x_2}, \dots, \mathbf{x_n}\}$, where each data is M-dimensional. In a GMM model, we need to determine the unknown distribution parameters ($w_i, \boldsymbol{\mu_i}$ and $\boldsymbol{\Sigma}_\mathbf{i}$) 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 $\boldsymbol{\Sigma}_\mathbf{i}$) 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 $\boldsymbol{\Sigma}_\mathbf{i}$) 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 (2018) 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).

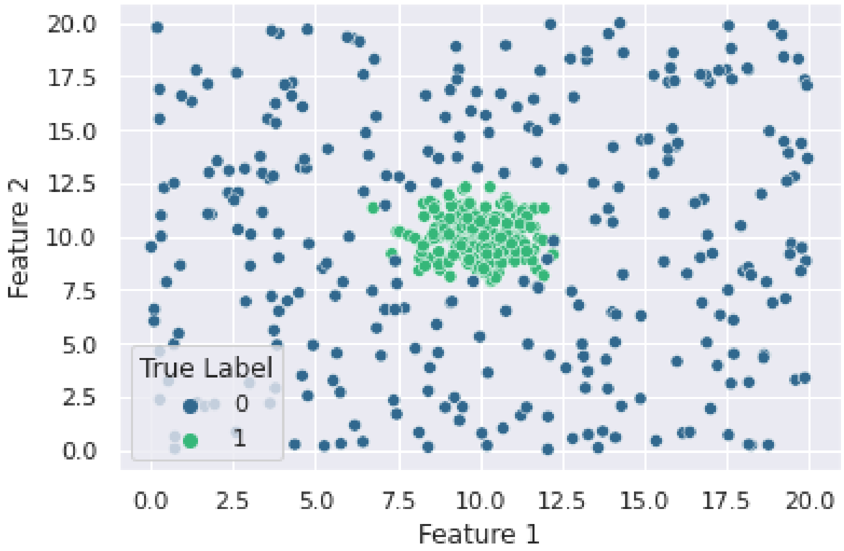


Figure 4.1: A Sample Simulated Dataset. Features 1 and 2 are just randomly generated features as an analogous of *pmra* and *pmdec*.

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).

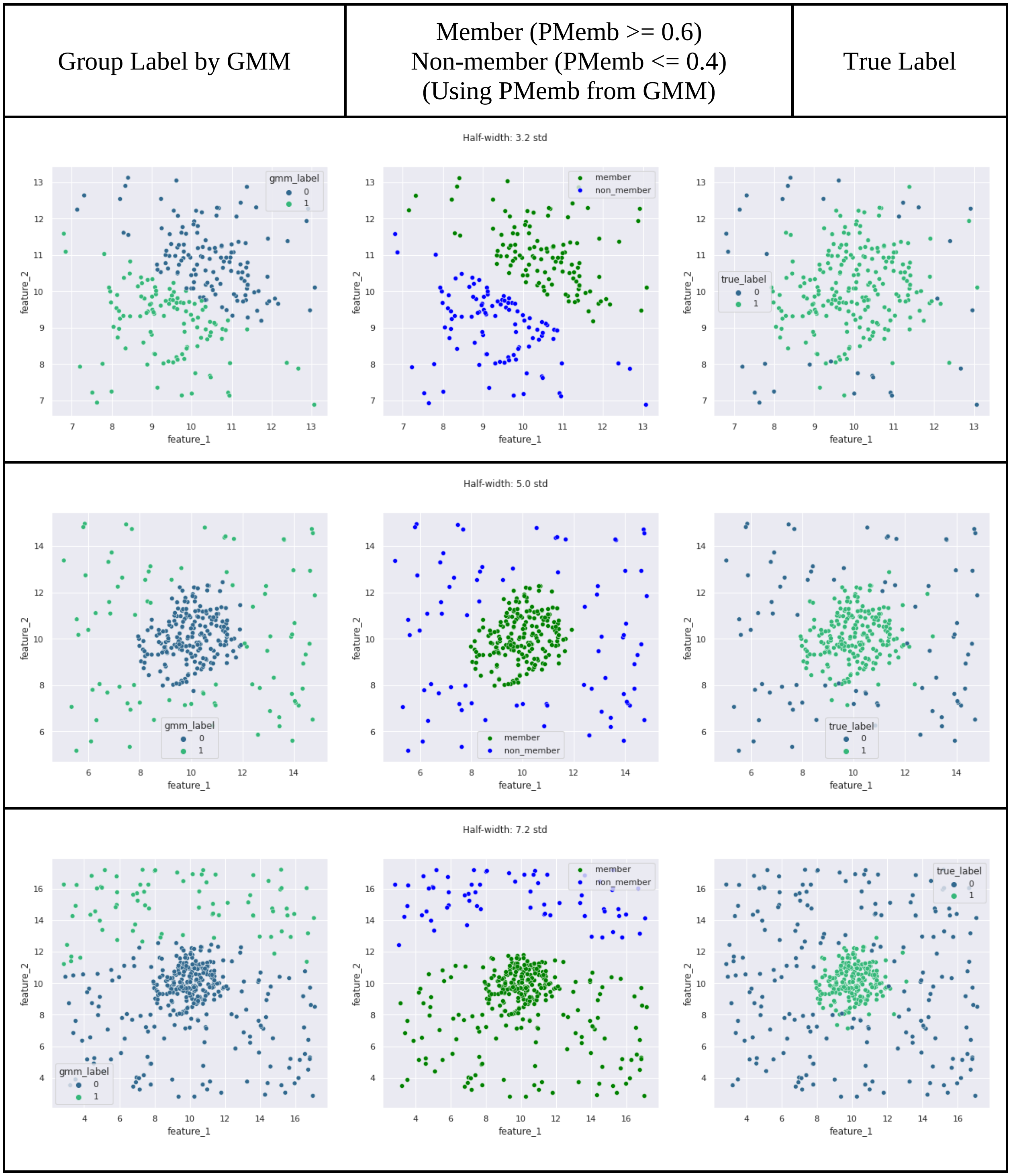


Figure 4.2: Performance of GMM for different value of half-width

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.

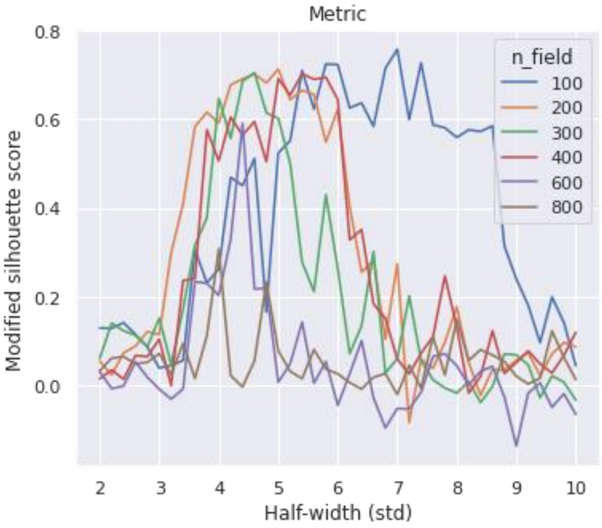


Figure 4.3: Performance of GMM for varying half-widths and for varying number of field stars.

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.

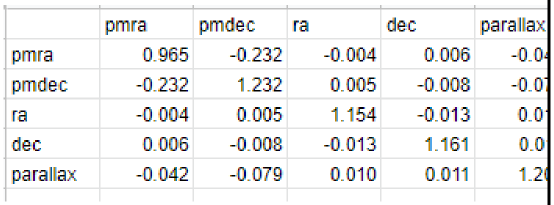
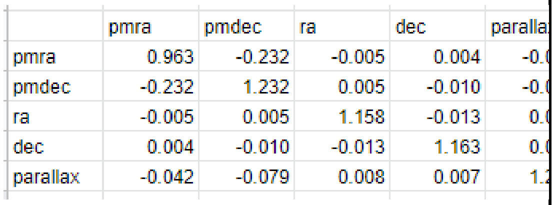
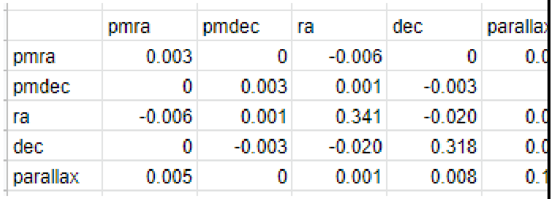
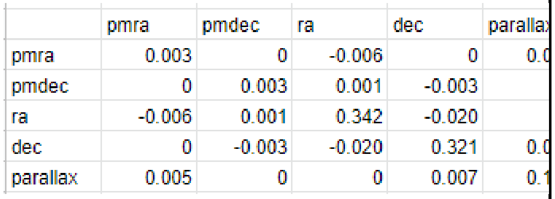
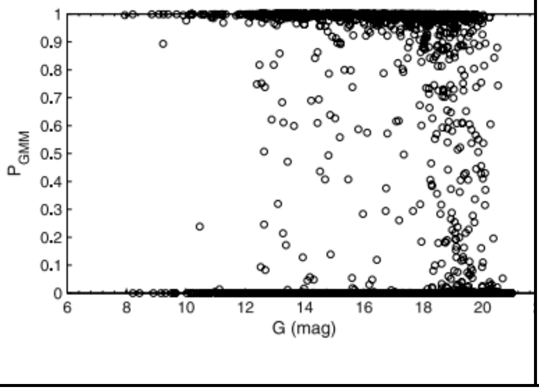
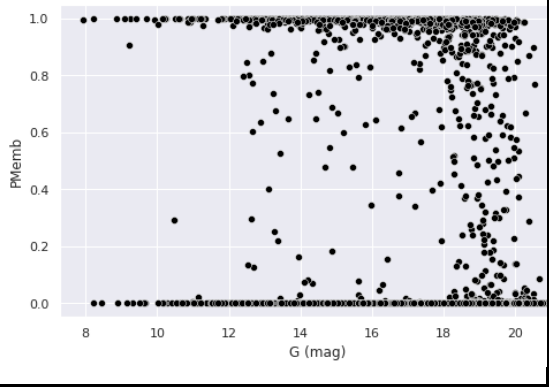
4.4. Past Paper Replication

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 (2018) 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. An open cluster typically has a diameter of 5-10 pc or smaller. But it is not uncommon for an open cluster to find some members (likely escapers) 10-20 pc away from the cluster center. At 860 pc, a physical size of 15 pc equals approximately 1 degree ((15/860) (206265/3600)1 deg). Thus a 1-degree search radius can be considered reasonable to start with for M67. Note that, for a cluster of a distance of 2000 pc or higher, 1 deg will be too large of an initial search radius as we would have a high ratio of field star to members.

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.

<table>
<tr><th></th><th>Gao M67 Paper (2018a)</th><th>Direct Replication</th></tr>
<tr><td>Sample Stars</td><td>7312</td><td>7316</td></tr>
<tr><td>n(PMemb >= 0.6)</td><td>1401</td><td>1414</td></tr>
<tr><td>n(PMemb >= 0.95)</td><td>1256</td><td>1260</td></tr>
<tr><td>n(PMemb <=0.0001)</td><td>5720</td><td>5693</td></tr>
<tr><td>Covariance matrix for field star, f</td><td>
<table>
<tr><th></th><th>pmra</th><th>pmdec</th><th>ra</th><th>dec</th><th>parallax</th></tr>
<tr><td>pmra</td><td>0.965</td><td>-0.232</td><td>-0.004</td><td>0.006</td><td>-0.04</td></tr>
<tr><td>pmdec</td><td>-0.232</td><td>1.232</td><td>0.005</td><td>-0.008</td><td>-0.07</td></tr>
<tr><td>ra</td><td>-0.004</td><td>0.005</td><td>1.154</td><td>-0.013</td><td>0.01</td></tr>
<tr><td>dec</td><td>0.006</td><td>-0.008</td><td>-0.013</td><td>1.161</td><td>0.0</td></tr>
<tr><td>parallax</td><td>-0.042</td><td>-0.079</td><td>0.010</td><td>0.011</td><td>1.20</td></tr>
</table></td><td>
<table>
<tr><th></th><th>pmra</th><th>pmdec</th><th>ra</th><th>dec</th><th>paralla</th></tr>
<tr><td>pmra</td><td>0.963</td><td>-0.232</td><td>-0.005</td><td>0.004</td><td>-0.0</td></tr>
<tr><td>pmdec</td><td>-0.232</td><td>1.232</td><td>0.005</td><td>-0.010</td><td>-0.0</td></tr>
<tr><td>ra</td><td>-0.005</td><td>0.005</td><td>1.158</td><td>-0.013</td><td>0.0</td></tr>
<tr><td>dec</td><td>0.004</td><td>-0.010</td><td>-0.013</td><td>1.163</td><td>0.0</td></tr>
<tr><td>parallax</td><td>-0.042</td><td>-0.079</td><td>0.008</td><td>0.007</td><td>1.2</td></tr>
</table></td></tr>
<tr><td>Covariance matrix for member star, m</td><td>
<table>
<tr><th></th><th>pmra</th><th>pmdec</th><th>ra</th><th>dec</th><th>parallax</th></tr>
<tr><td>pmra</td><td>0.003</td><td>0</td><td>-0.006</td><td>0</td><td>0.0</td></tr>
<tr><td>pmdec</td><td>0</td><td>0.003</td><td>0.001</td><td>-0.003</td><td></td></tr>
<tr><td>ra</td><td>-0.006</td><td>0.001</td><td>0.341</td><td>-0.020</td><td>0.0</td></tr>
<tr><td>dec</td><td>0</td><td>-0.003</td><td>-0.020</td><td>0.318</td><td>0.0</td></tr>
<tr><td>parallax</td><td>0.005</td><td>0</td><td>0.001</td><td>0.008</td><td>0.1</td></tr>
</table></td><td>
<table>
<tr><th></th><th>pmra</th><th>pmdec</th><th>ra</th><th>dec</th><th>parallax</th></tr>
<tr><td>pmra</td><td>0.003</td><td>0</td><td>-0.006</td><td>0</td><td>0.0</td></tr>
<tr><td>pmdec</td><td>0</td><td>0.003</td><td>0.001</td><td>-0.003</td><td></td></tr>
<tr><td>ra</td><td>-0.006</td><td>0.001</td><td>0.342</td><td>-0.020</td><td></td></tr>
<tr><td>dec</td><td>0</td><td>-0.003</td><td>-0.020</td><td>0.321</td><td>0.0</td></tr>
<tr><td>parallax</td><td>0.005</td><td>0</td><td>0</td><td>0.007</td><td>0.1</td></tr>
</table></td></tr>
<tr><td>PMemb vs G-band magnitude</td><td>
</td><td>
</td></tr>
</table>

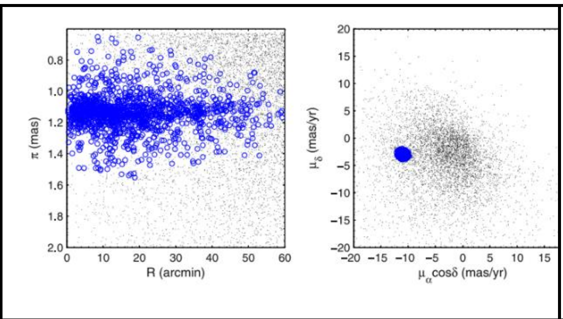
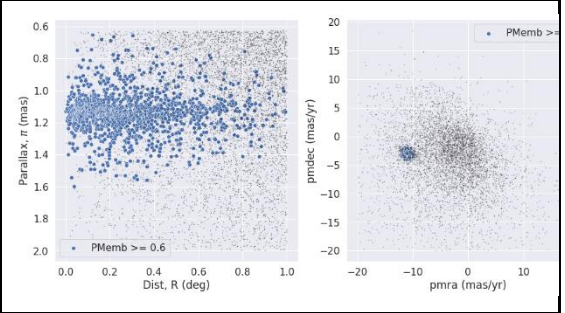
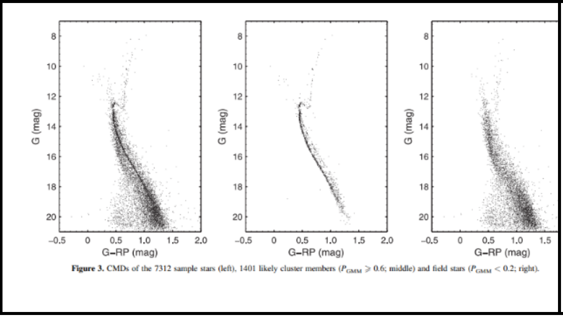
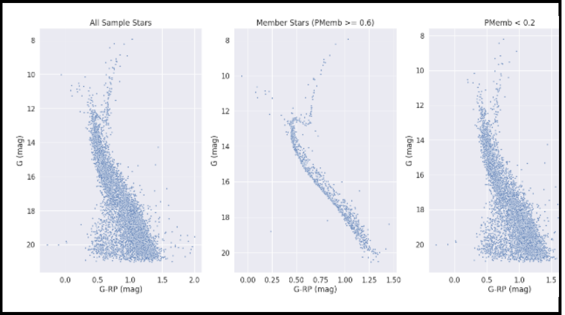
Parallax vs distance from the cluster center, proper motion plot		
Color Magnitude Diagrams (CMD)	 **Figure 3.** CMDs of the 7312 sample stars (left), 1401 likely cluster members ($P_{\mathrm{GMM}} \geq 0.6$; middle) and field stars ($P_{\mathrm{GMM}} < 0.2$; right).	

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.

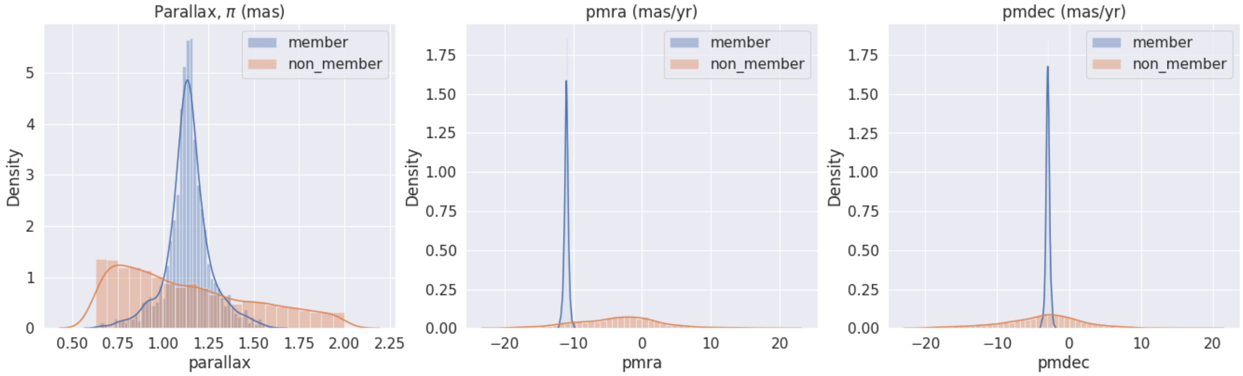


Figure 4.4: Distributions of the identified member and non-member for M67.

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 the similar peak position and similar distribution. Moreover, 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.

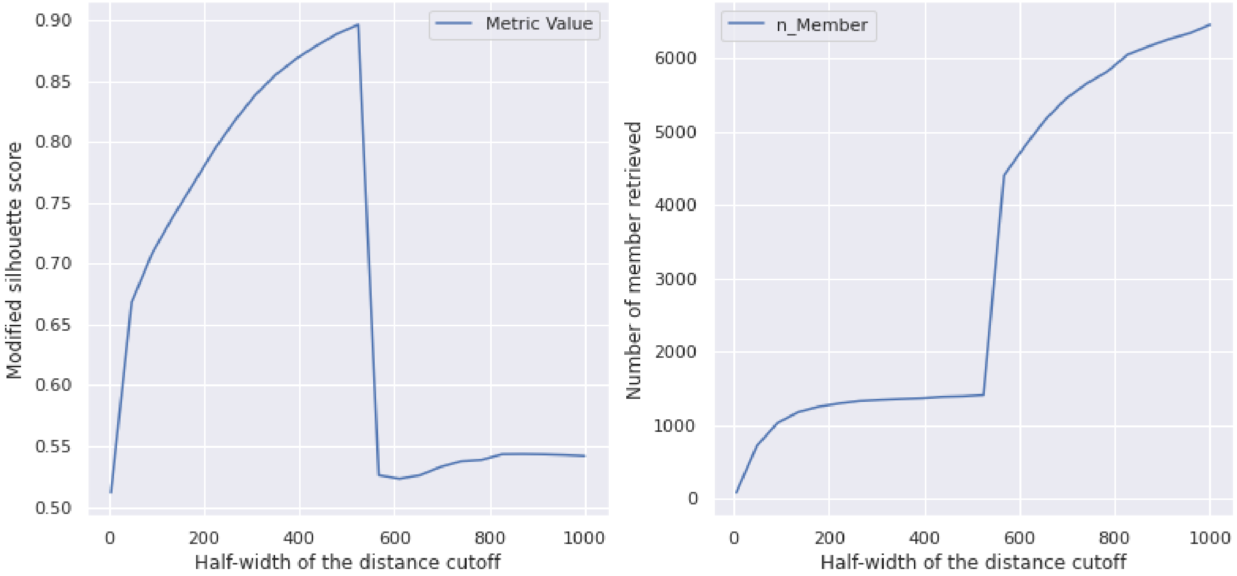
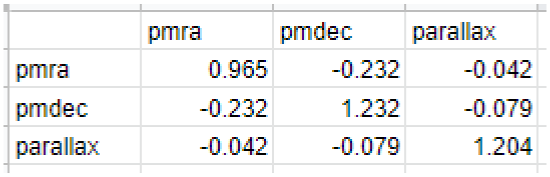
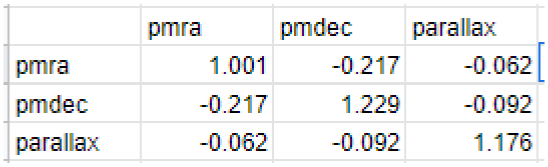
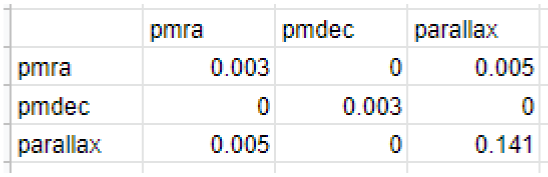
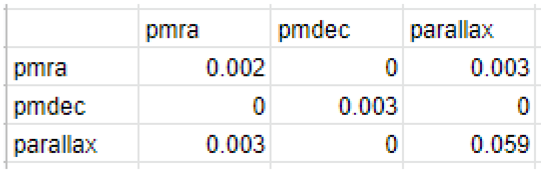
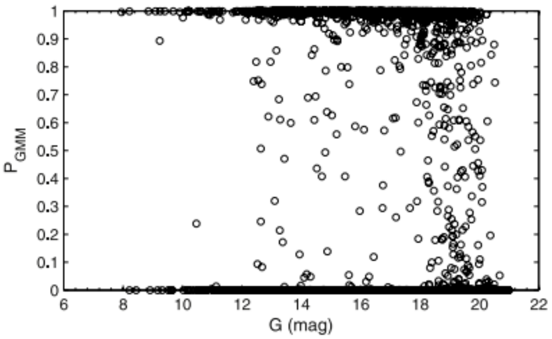
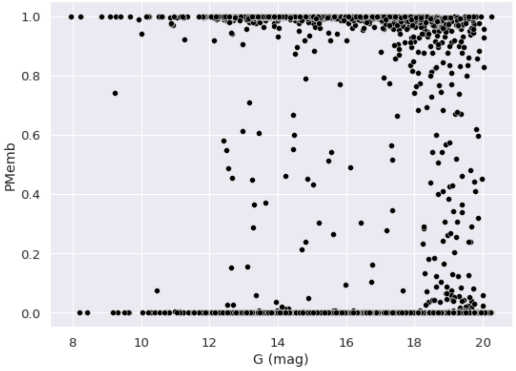


Figure 4.5: MSS score for varying half-widths for distance which we can use to choose an optimum distance cutoff.

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.

<table>
<tr><th></th><th>Gao M67 Paper (2018)</th><th>Modified Replication</th></tr>
<tr><td>Sample Stars</td><td>7312</td><td>6118</td></tr>
<tr><td>n(PMemb >= 0.6)</td><td>1401</td><td>1394</td></tr>
<tr><td>n(PMemb >= 0.95)</td><td>1256</td><td>1274</td></tr>
<tr><td>n(PMemb <=0.0001)</td><td>5720</td><td>4558</td></tr>
<tr><td>Covariance matrix for field star, Σ_f</td><td>
<table>
<tr><th></th><th>pmra</th><th>pmdec</th><th>parallax</th></tr>
<tr><td>pmra</td><td>0.965</td><td>-0.232</td><td>-0.042</td></tr>
<tr><td>pmdec</td><td>-0.232</td><td>1.232</td><td>-0.079</td></tr>
<tr><td>parallax</td><td>-0.042</td><td>-0.079</td><td>1.204</td></tr>
</table></td><td>
<table>
<tr><th></th><th>pmra</th><th>pmdec</th><th>parallax</th></tr>
<tr><td>pmra</td><td>1.001</td><td>-0.217</td><td>-0.062</td></tr>
<tr><td>pmdec</td><td>-0.217</td><td>1.229</td><td>-0.092</td></tr>
<tr><td>parallax</td><td>-0.062</td><td>-0.092</td><td>1.176</td></tr>
</table></td></tr>
<tr><td>Covariance matrix for member star, Σ_m</td><td>
<table>
<tr><th></th><th>pmra</th><th>pmdec</th><th>parallax</th></tr>
<tr><td>pmra</td><td>0.003</td><td>0</td><td>0.005</td></tr>
<tr><td>pmdec</td><td>0</td><td>0.003</td><td>0</td></tr>
<tr><td>parallax</td><td>0.005</td><td>0</td><td>0.141</td></tr>
</table></td><td>
<table>
<tr><th></th><th>pmra</th><th>pmdec</th><th>parallax</th></tr>
<tr><td>pmra</td><td>0.002</td><td>0</td><td>0.003</td></tr>
<tr><td>pmdec</td><td>0</td><td>0.003</td><td>0</td></tr>
<tr><td>parallax</td><td>0.003</td><td>0</td><td>0.059</td></tr>
</table></td></tr>
<tr><td>PMemb vs G-band magnitude</td><td>
</td><td>
</td></tr>
</table>

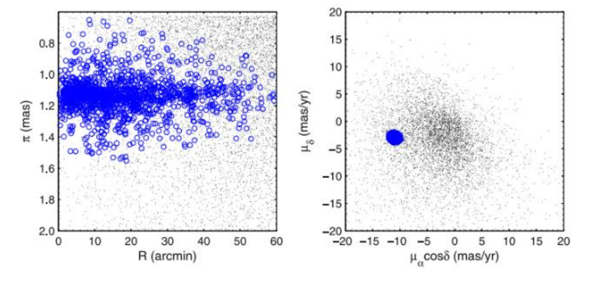
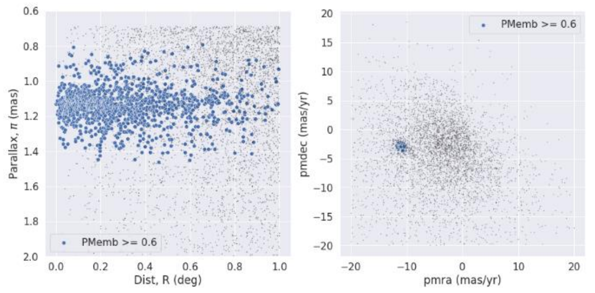
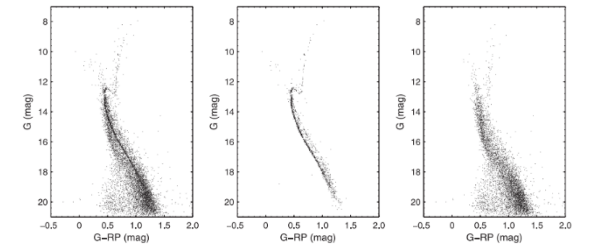
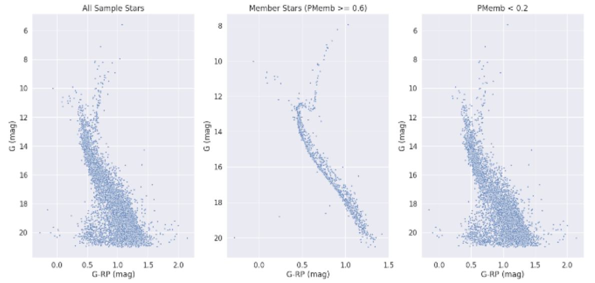
Parallax vs distance from the cluster center, proper motion plot		
Color Magnitude Diagrams (CMD)	 **Figure 3.** CMDs of the 7312 sample stars (left), 1401 likely cluster members ($P_{GMM} \geq 0.6$; middle) and field stars ($P_{GMM} < 0.2$; right).	

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.

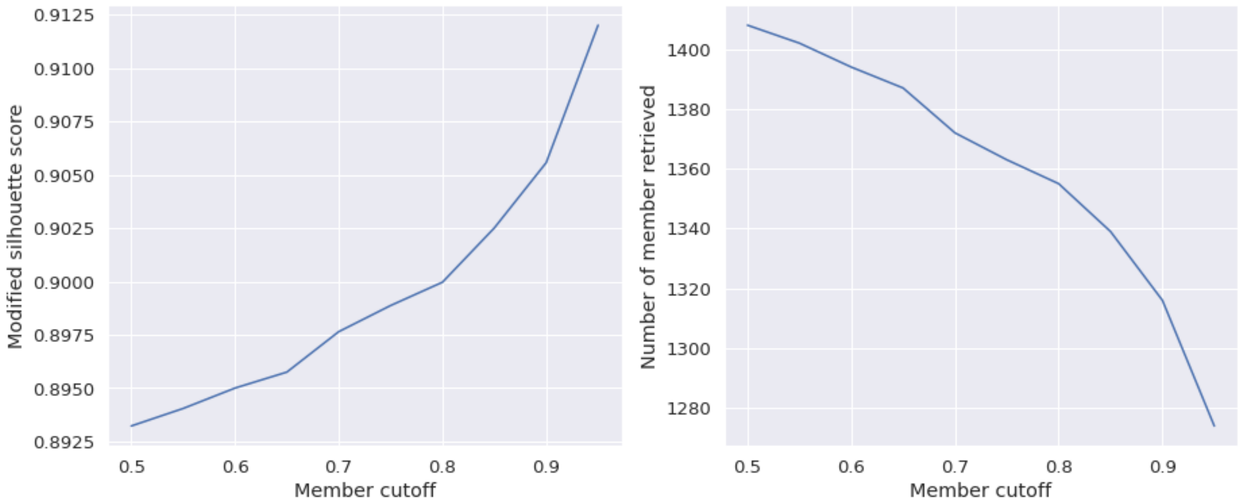


Fig 4.6 MSS vs Member cutoff to determine an optimum cutoff for modified replication of M67

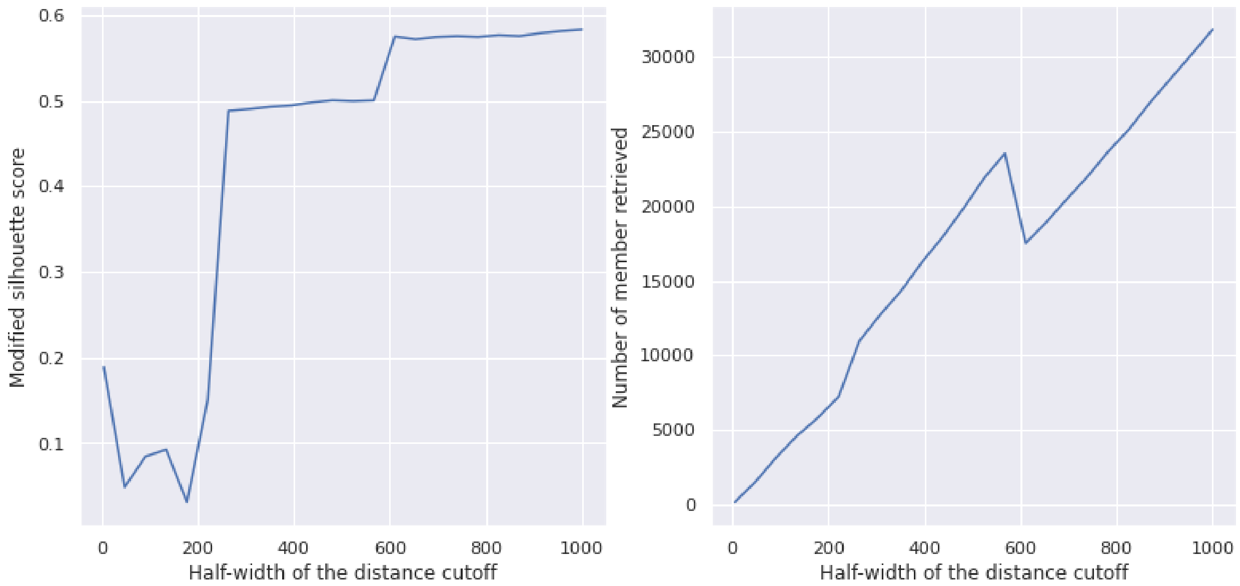
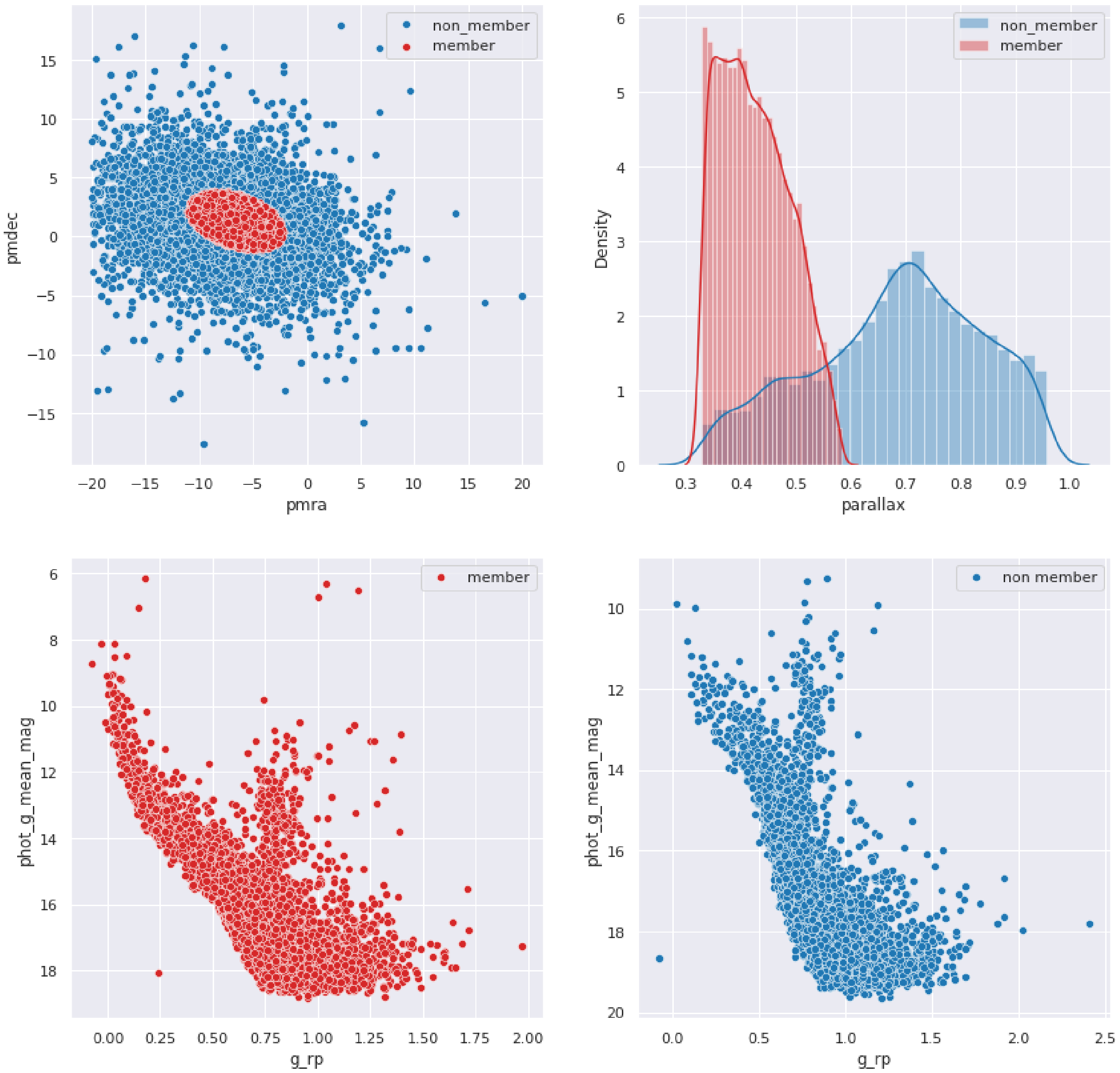


Figure 4.7: half-width vs MSS plot to find the optimal distance cutoff empirically for the new cluster NGC 3766.

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. We can see that, compared to the field stars, the member stars are more closely stays together in the proper motion space as well as in the parallax. We received a maximum MSS score of 0.613 using this model, which is lower than the modified replication. The color magnitude diagram of the member follow the theoretical CMD, though the thickness of the CMD is larger than ideal.



MSS Metric Value: 0.613

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 create 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 $f \leq t$, they 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 misclassify that observation. Suppose we have k different classes $(1,2,3 \ldots, k)$ in node m and the number of observations for each class are denoted by $N_1, N_2, \ldots, N_k$ 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,

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

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

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.

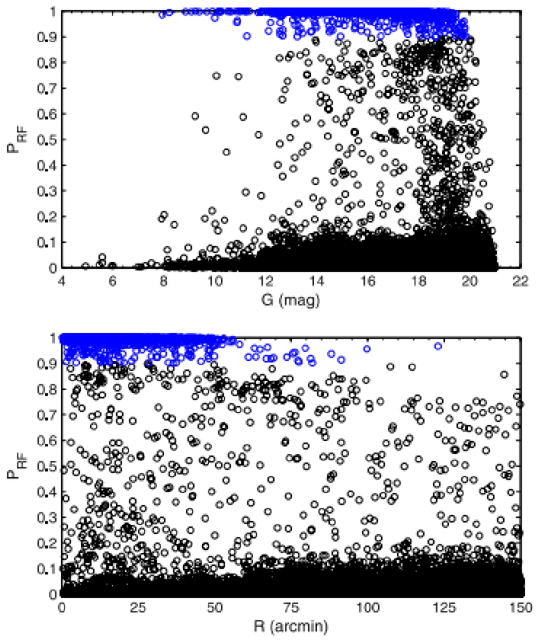
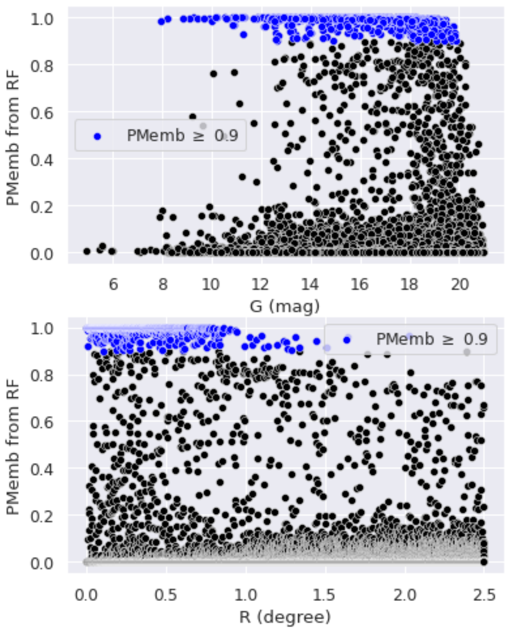
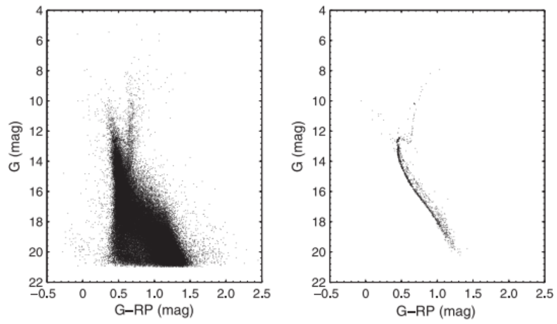
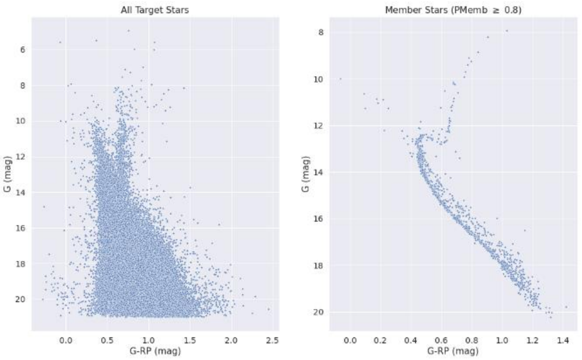
One important point to note is that the random forest still predicts using the piecewise functions generated by all those trees. Thus, random forest is still cannot work well for extrapolation. We need to ensure that the range of the new dataset features, where we want to apply Random Forest model, does 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 (2018) 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 (2018) paper are compared in Table 5.1.

	Gao M67 Paper (2018)	Direct Replication
Target Stars	71,117	70987
n(PMemb >= 0.6)	1502	1528
n(PMemb >= 0.95)	1361	1377
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 ($P_{RF} \geqslant 0.8$; right).	

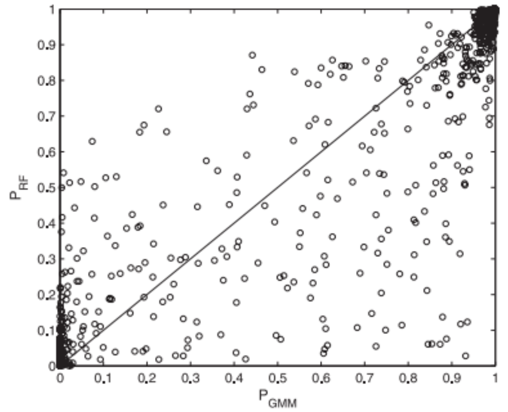
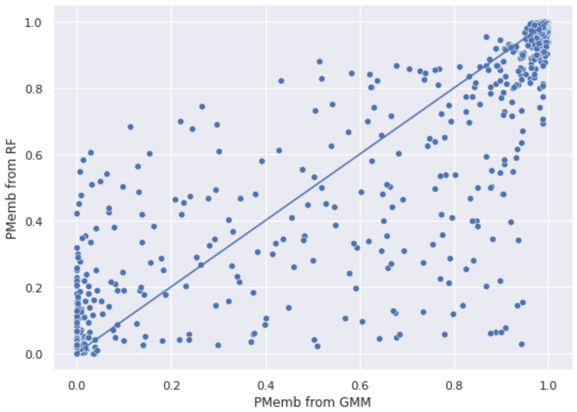
Correlation between both of the PMemb	 **Figure 8.** A comparison between the GMM and RF membership probabilities of the 7312 sample stars.	

Table 5.1: Comparison between Gao (2018) 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

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.

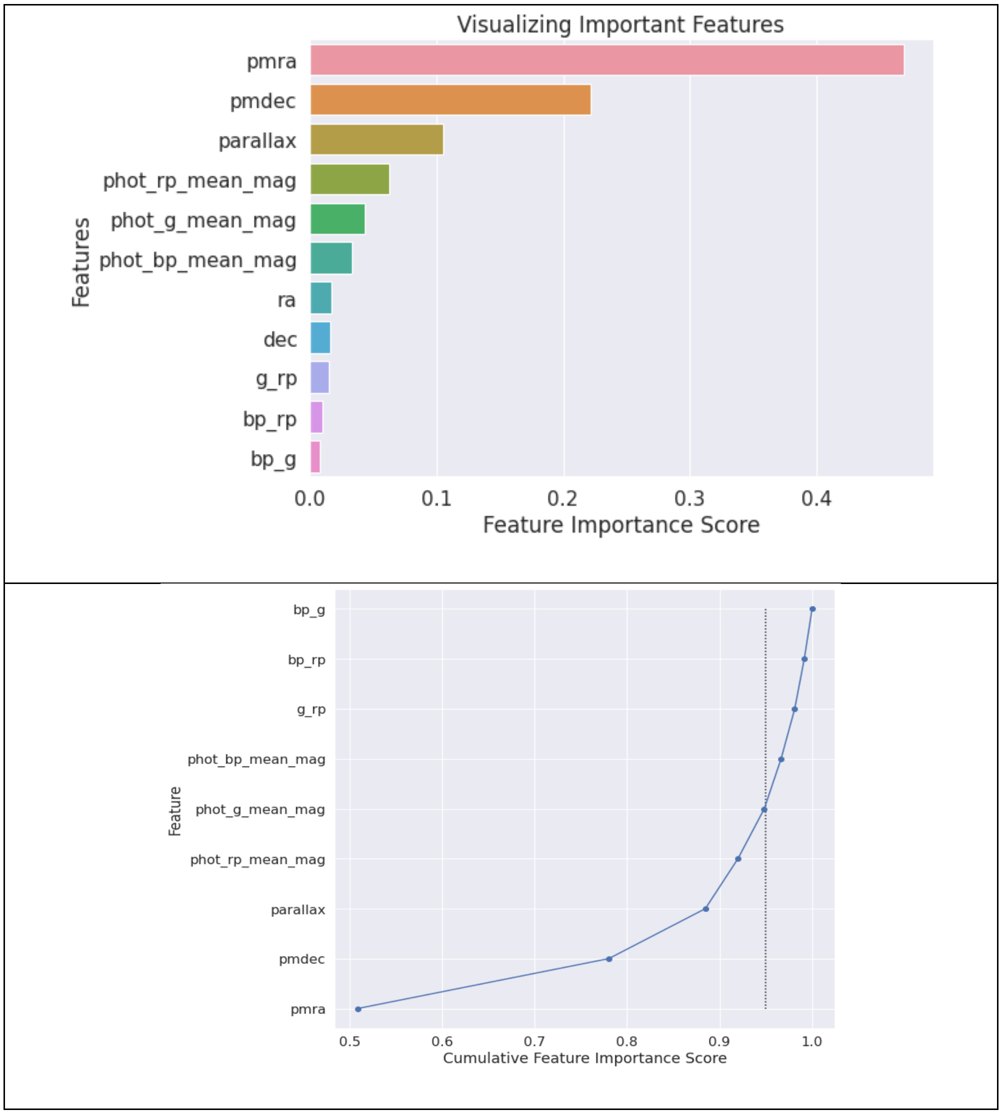


Fig 5.1: Feature importance of RF model using mean decrease in impurity (MDI)

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, that 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, $\overline{s_f}$ with randomly shuffled feature *f*. The PFI of feature *f* for the given model is defined as,

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

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,

$$\begin{aligned} SHAP_f &= \sum_{S\subseteq N\setminus\{f\}} p_{S+\{f\}} \times MC_{f,\{S+\{f\}\}} \\ &= \sum_{S\subseteq N\setminus\{f\}} \frac{|S|!\,(|N|-|S|-1)!}{|N|!} \times (v_{S\cup\{f\}} - v_S) \end{aligned}$$

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.

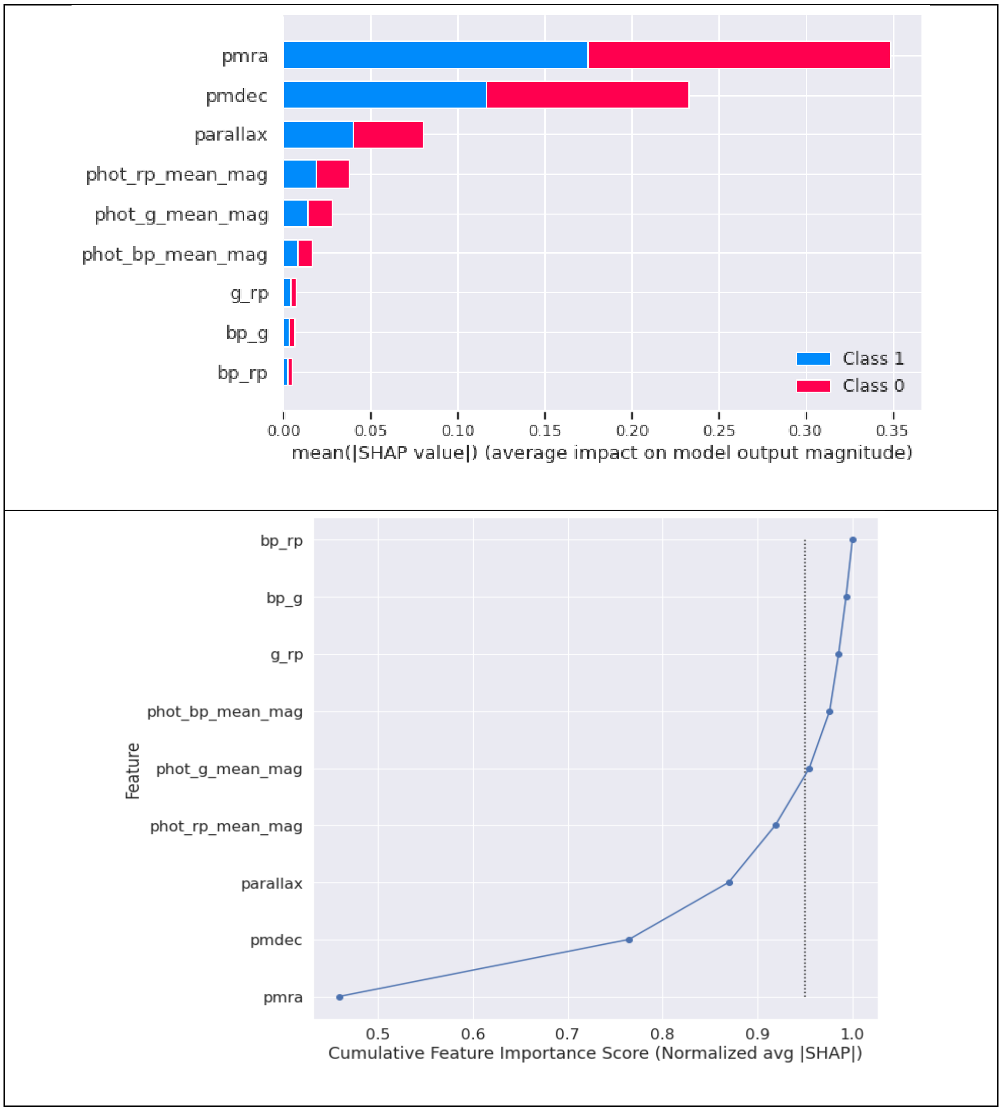


Fig 5.2: Feature importance of RF model using normalized average of absolute SHAP values for all the features.

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 a 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 parameter. We can look this as an optimization problem. Ideally, we want to choose a set of model parameters 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 variable we want to optimize. The parameters that we change are the model parameters. We can do cross validation with randomized search to solve this problem (Bergstra & Bengio, 2012). 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.

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
bootstrap (Whether bootstrapping the samples)	True, False
ccp_alpha (Complexity parameter)	$0, 2^{-10}, 2^{-9}, 2^{-8}, 2^{-7}, 2^{-6}, 2^{-5}, 2^{-4}, 2^{-3}, 2^{-2}, 2^{-1}, 2^{0}$

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

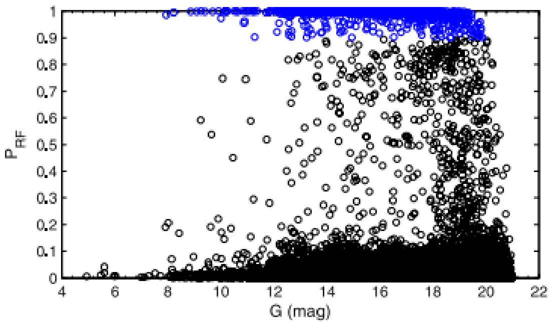
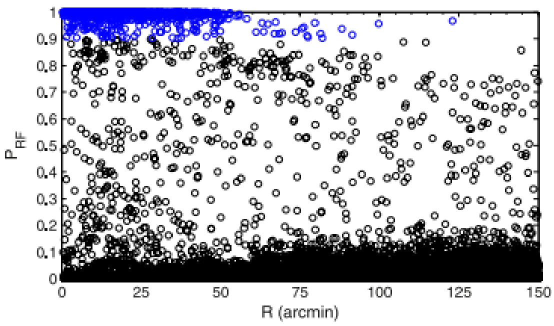
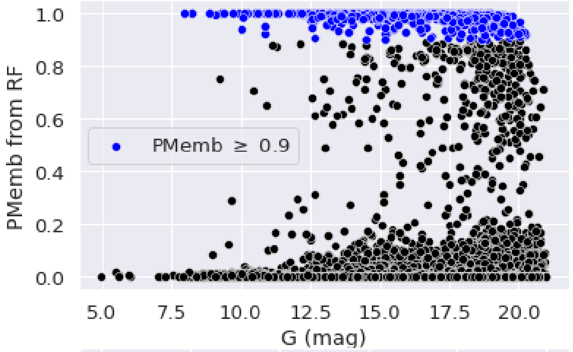
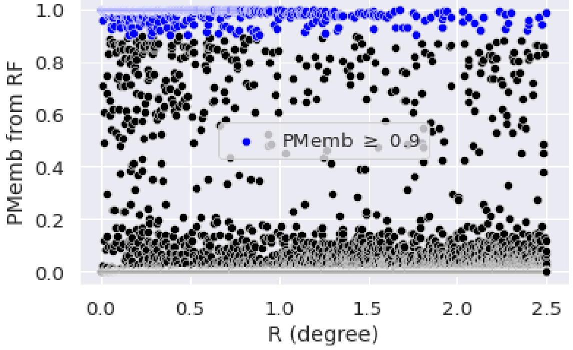
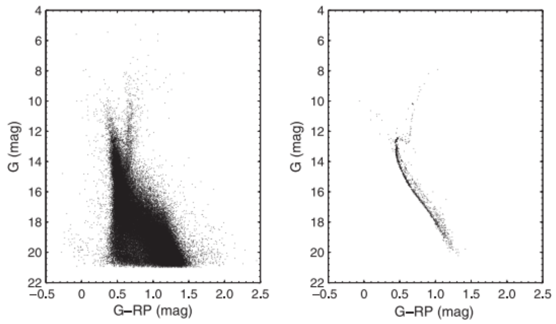
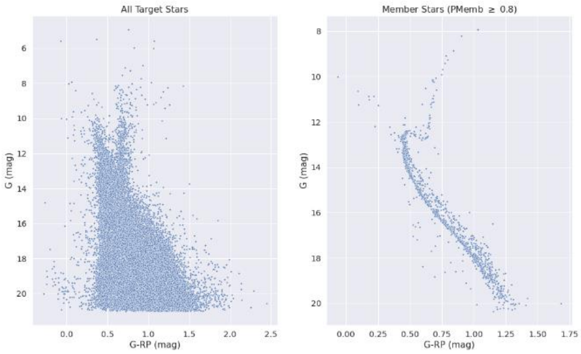
We used the grid presented in Table 5.2 to randomly select 100 models with different model parameter values to build a 5-fold cross validation model. Then we chose the best model among those 500 iterations. The best parameters found in this process are compared with the Gao paper's model parameter in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

Model Parameters	Gao Paper (2018)	Modified Paper (2018) (Optimized by cross validation)
n_estimators (Number of decision trees)	7000 (chosen)	2000
max_features (Number of features in each tree)	4 (chosen)	5
max_depth (Maximum depth of the tree)	None (default)	20
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
bootstrap (Whether bootstrapping the samples)	True	True
ccp_alpha (Complexity parameter)	0 (default)	0

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

The overall result from the modified replication is presented in Table 5.3. We can see that the number of member stars are increased in this model in addition to the stronger justifications of the model.

	Gao M67 Paper (2018)	Modified Replication

Target Stars	71,117	70,987
n(PMemb >= 0.6)	1502	1618
n(PMemb >= 0.95)	1361	1492
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 ($P_{RF} \geqslant 0.8$; right).	

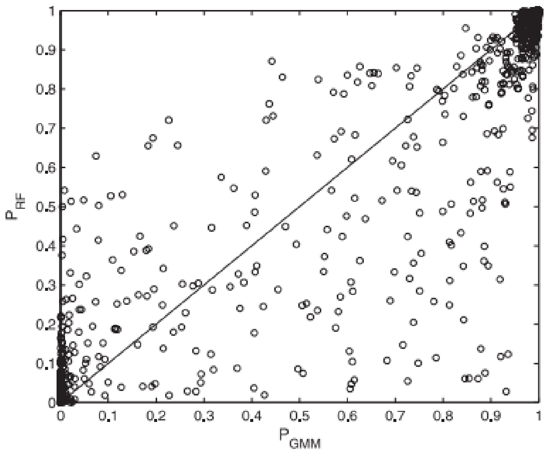
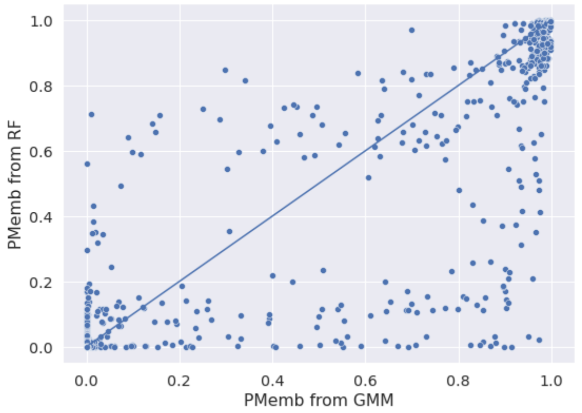
Comparison between the PMemb of both models.	 **Figure 8.** A comparison between the GMM and RF membership probabilities of the 7312 sample stars.	

Table 5.3: 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.

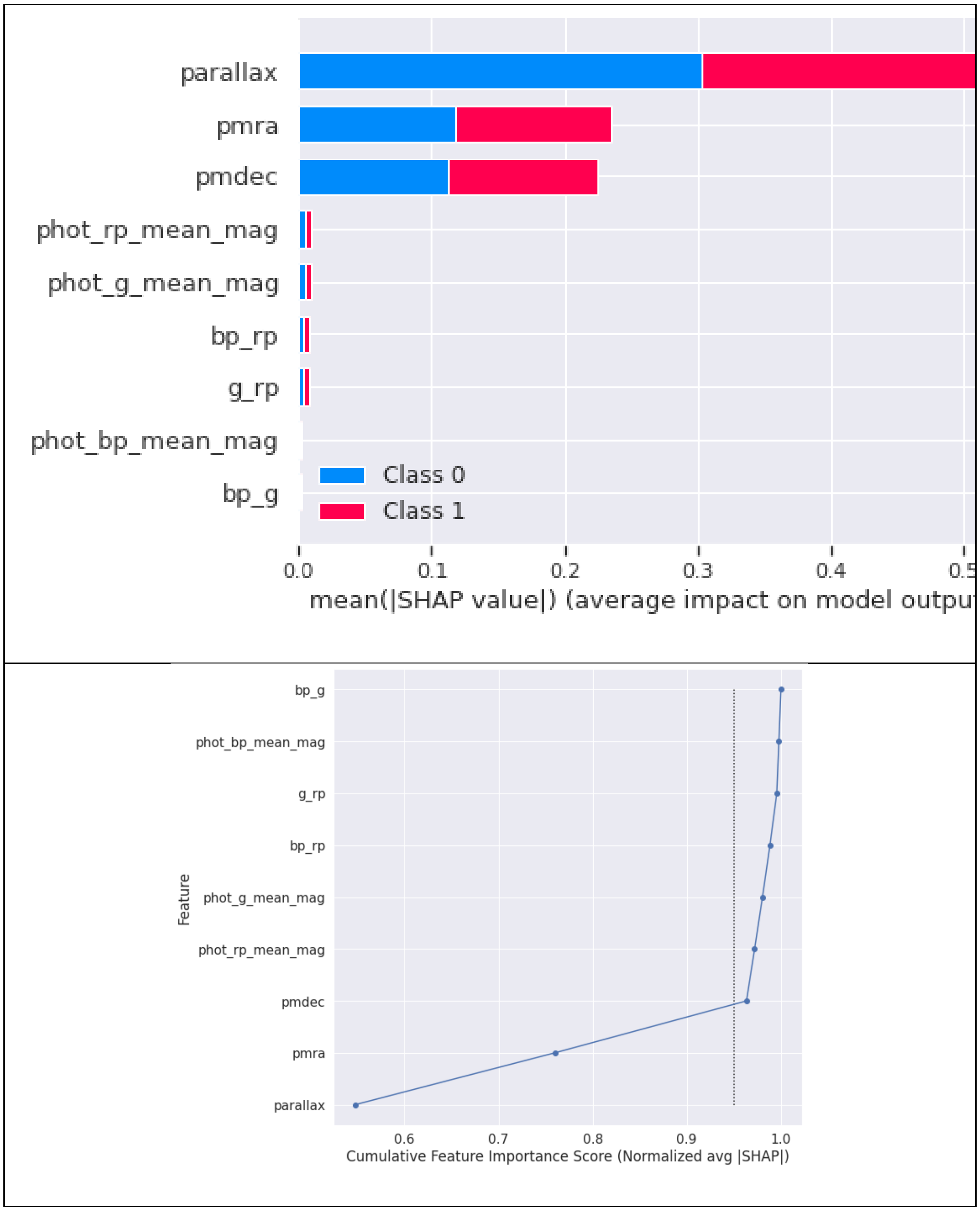
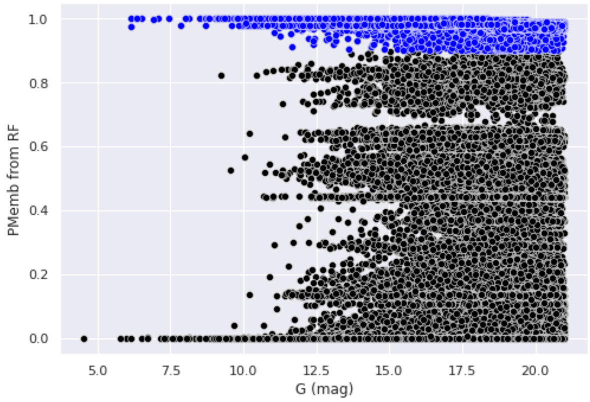
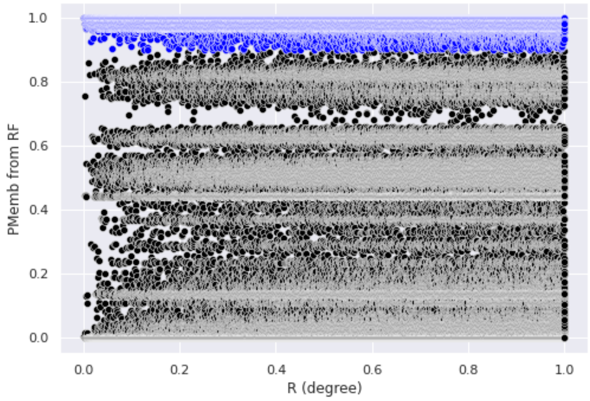
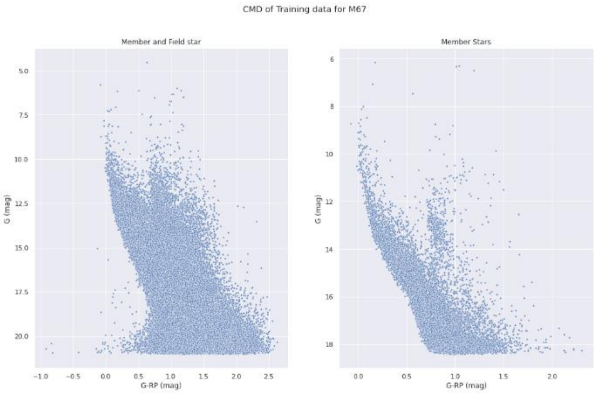


Fig 5.3: Feature importance of RF model using normalized average of absolute S

Based on 95% threshold, we only found pmra, pmdec and parallax as the important parameter and used them in RF model. Then we applied the RF model in the larger region (1 degree radius). The results are summarized in Table 5.4.

Target Stars	5,48,312
PMemb >= 0.8	2,45,304
PMemb = 1	26,058
PMemb vs G Mag, PMemb vs R (distance from the cluster center)	 
CMD	

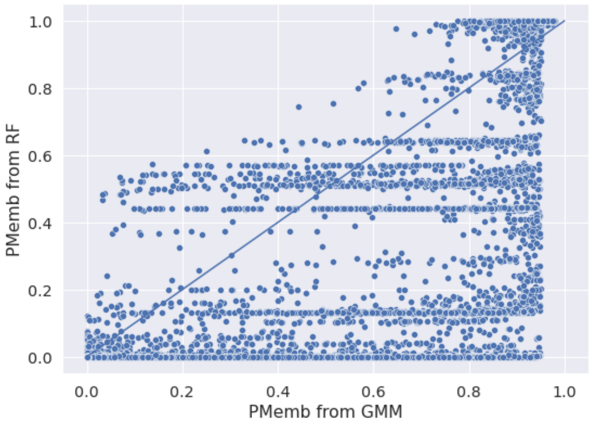
Comparison between the PMemb of both models.	

Table 5.

6. Conclusion

In this paper, we just go over two example of ML model. Using the metrics defined in Sec 3.3., we can compare different unsupervised and supervised models to find the most optimal one for this problem. The discussions of section 4 and 5 will help us to decide and justify the best practices while applying these two type of models.

Appendix: Code Notebook

Understanding GMM:

https://colab.research.google.com/drive/1KV2AH3_OKG4CKTHbRrMZZAdCembTdDu-?usp=sharing

GMM Replication:

https://colab.research.google.com/drive/13tZo3n-UDRceBSWuJs3svxBhWcavkBpZ?usp=sharing

Modified Replication:

https://colab.research.google.com/drive/1-Ms5aF2_1hjHqrawwcumiu81-ZmqYLGV?usp=sharing

New Model:

https://colab.research.google.com/drive/1CVXusacaMxIMC9LjNtElCr3nuuf1_xBP?usp=sharing

Future Work Plan

Work	Deadline
Continue working on HC/LO justification (phase 1)	03 March 2022
Read again with fresh eye and add more justifications, especially in conclusion section	03 March 2022
Finish code commenting	07 March 2022
Prepare slide for oral defense	07 March 2022
Work on the feedback from oral defense meeting	10 March 2022
Add more explanation on the running RF in new model section and background section	15 March 2022
Revise the work on HC/LO (phase 2)	18 March 2022
Meet with advisor (and if possible, with second grader) to discuss on feedback on this assignment and to discuss on any remaining question	20 March 2022
Send to a peer for feedback	22 March 2022
Final proofreading and polishing	27 March 2022

HC and LO Justification

LO	Actual Applications	Intended Applications
#cs156- #modelmetrics*	In the evaluation metric section, I first defined the potential metrics that I can use for this problem. Then I compared between the metrics, showing why certain metrics cannot work well in this particular setting based on their shortcomings and their underlying assumptions. I also chose a metric based on the context of the problem and the study area (i.e., I explained why we want to minimize false positive, then chose precision as it does minimize false positive). Lastly, I also proposed a new metric for unsupervised model considering the structure of our dataset.	The justification that the new metric is effective can be further improved by simulating some dataset with good and bad separation between member and field stars and check how the new metric can provide higher value for good separation and lower value for bad separation.
#cs156-	In the hypothesis section, I	In the application of GMM and

#unsupervisedlearning*	mentioned why we need to use an unsupervised model (as we do not usually have labels in the real data) and how it can be combined with a supervised method to improve our membership detection. I also mentioned the necessity of comparing different models to find the best working model. In the workflow section, I further mentioned how unsupervised model can use features of the new data to generate the labels for them. Then I discussed the constraint of having a continuous variable as our outcome and how we can use that constraint to chose or modify specific unsupervised model.	UPMASK, I will further discuss the nuances of unsupervised learning algorithms. For example, how I need to preprocess and filter my data depending on the assumptions of the model, how can I choose the model parameters, how to interpret the output of the model etc.
#CS156- #Overfitting		While applying the ML models, especially the supervised models, I will explain when and why a

		model can overfit and what measures we can take to avoid that. For example, a single Decision Tree can overfit the data specially if we do not choose the model parameters (depth, maximum leaf etc.) effectively. A random forest can decrease the bias using a lot of trees to increase the variance. Also, we can use cross-validation to optimize the model parameters to avoid overfitting. Lastly, we can compare the metric value for test and train subset to check if there is any overfitting.
#CS156- #regressionalgorithm		I will implement two of the regression supervised models: random forest and support vector machine. We need regression here because our output is a continuous variable. I also need

		to justify why I chose these two models as the first two models to solve this problem. I need to explain the underlying assumptions of these models and how they work in this particular problem.
#CS110- #PythonProgramming*	I used python to collect the dataset, preprocessing, filtering, and implementing the GMM model. I try to vectorize using NumPy array and pandas whenever applicable. I also generated meaningful plots using seaborn and matplotlib.	I still need to work on to improve the efficiency of some of the algorithms and functions that I used. I will look if I can use more useful data structure or function to reduce the computational time and to avoid writing repeating codes.
#CS110- #CodeReadability	I used appropriate variable names, function names and consistent naming conventions. The code blocks are organized in a way so that it can be understandable by external readers.	I need to add docstrings to my functions. I will also add more comments to make sure that external readers can follow through the notebook. If possible, I will also try to include useful error messages.

#NS110U- #conservationandmotion		In the background section, I can discuss how the conservation of energy and momentum can be used to determine the typical physical radius of an open cluster considering their mass.
NS110U #newtoniangravity		In the background section, I can explain how the Newtonian gravity is working to hold together the open clusters by their mutual gravity. Also, I can mention why they are loosely bound considering their mass and why and how they evaporate (i.e., move away) from the cluster eventually after a certain time limit.
NS110U #newtonslaws		In the background and discussion section, I can discuss how we can use newton's laws of motion to approximate the motion of the stars in a star cluster and how

		they can eventually move towards the center or move away from the cluster.

HC	Actual Applications	Intended Applications
#variables*	In workflow sections, I identified the input variables (features or independent variable) for our problem and our interested response variable (dependent variable or output). 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). Then considering this constraint, I explained how we can modify our choice of unsupervised	I need to discuss if there are any more input variables in the system (i.e., open cluster) that can be used to determine the membership probability and if we have data available for that. Also, while implementing the ML models, I can discuss the parameters of the model, which is different than the variables. I can also mention how we can optimize these parameters.

	model.	
#gapanalysis*	In the background and hypothesis section, I mentioned some of the works that have been done on the membership determination problems. In this problem, our initial state is that due to GAIA data very recently we get precise astrometric data, and our goal state is to find confident members for the open clusters. The available solution is that people are using different ML models to solve the problem. I identified some of the existing gaps in this solution (cannot use supervised directly or cannot justify which model is better). Then I propose a plan to fill in this gap.	
#hypothesisdevelopement*	In the hypothesis section, I first showed evidence from the past	

	papers 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 this evidence, I predicted that the combined model would result in a better membership determination. I also explained what I mean by better qualitatively and quantitatively. Lastly, I also propose a workflow and evaluation metric to test that hypothesis.	
#plausibility	While building the hypothesis, I also had to check the plausibility of the underlying assumptions of the hypothesis. In the workflow	I need to mention if there are any other underlying assumptions behind the current hypothesis and check if that is

	section, I showed how I am planning to combine both unsupervised and supervised methods considering the response variable. So, it is possible to make them work together. Also	plausible based on the knowledge of the field.
#evidencebased	In order to base my hypothesis on the proper evidence, I mentioned the papers on ML literature and the papers which used a combined method.	In every choice that I need to make while implementing the models (i.e. how to filter, what threshold to choose, what will be the parameter value) I need to provide proper evidence either from the knowledge and reasoning of the field or from the proper authoritative sources.
#studyreplication*	For each of the models, I replicated at least one existing paper that used that model. The main reason behind the replication is to understand the implementation of the model	

	better. I need to compare the findings and plots of my replication with the original paper and explain any discrepancies or similarities.	
#designthinking*	While implementing the GMM 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 had to take feedback from professors and peers and point experts to understand what can be improved (feedback incorporation).	In the reflection section, I will talk about the changes that I need to make on my models based on the feedback from others. I will add what went wrong and how did I understand that it went wrong and how did I modify the code to its final version.
#dataviz		While representing or interpreting the results of the ML models or justifying any of

		my choices with simulated data, I need to use proper visualizations of meaningful metrics. I need to discuss why I chose such visualizations and what can we learn from there. The caption, axis labels, legend, and color-coding should also follow the academic standard.
#selfawareness		This will be used in the process of the project and written in the reflection section. I need to understand myself first in terms of how good I am to fulfill each small deadline and to keep myself accountable. If I am not good at that, then I need to take help from my peers or friends to keep myself accountable. I also need to keep track of my time and share my goals with other people to increase

		accountability.
#algorithm		While implementing all the ML models, I need to come up with an efficient and effective algorithm. I need to write commented codes with clear steps and justify any choices that I make along the way (i.e., choice of the parameter value, data structure etc.)
#composition		I need to polish my writings to make them more parsimonious by cutting down unnecessary words and avoiding repetitions. Also, I need to make sure that the paragraphs discuss a specific topic and the transitions between paragraphs and sections are smooth.
#audience		I need to keep in mind that my target audience is my peers who

		have taken the same sort of courses. So, I need to avoid complex technical or astronomical jargon. If it is necessary, then I need to provide a definition first. Also, as my target audience is peers, I should share my writings with them to get feedback on if it is understandable enough.

Glossary

Bayesian Inference: Bayesian inference uses the bayes equation to derive the posterior:

P(|data, model)P(data| , model) P(| model).

where P (data| , model) is the likelihood, means the probability density using a specific parameter(s) for an observed data and P(| model) is called the prior or initial educated guess for the parameter distribution and P(|data, model)is called the posterior distribution for our parameter, which we want to derive or estimate.

Maximum Likelihood: In the Maximum Likelihood model, we try to find the parameter for a distribution which will give the maximum probability density for the observed data.

K-means Clustering: In k-means clustering we randomly select k *center points* in the n-dimensional space, where n is the number of our input variables. Then we assign each data point to the clusters of the *center point* from which the Euclidean distance is minimum. Once we find k clusters, we find the mean position of each star and set them as new *center points*. We repeat this until the center points become fixed and calculate the summation of the variance for each of the clusters.

While applying k-means clustering in the star cluster problem, we calculated the closeness of each star using the Euclidean distance between them and based on the closeness (or distance) we grouped the full dataset into k different clusters. Here, due to the similarity of the astrometric data for the cluster members they will be very close to each other while the field stars will be in a random position.

Spectral Clustering: Spectral clustering used the similar idea as in k-means clustering but instead of calculating the Euclidean distance in n-dimensional space, it creates graphs while connecting the data points based on the similarity of the input variables and group the datasets into separate closely connected clusters.

Random Forest: Random Forest is based on the decision tree approach, where the training dataset is used to build a tree, which later can classify or predict the output of a new data. Random forest, to reduce the variance of a single tree, builds many trees each time using a random subset of the data and random subset of the input variables. Then the final prediction or classification is the average or the majority of all such trees. As a supervised method, random forest needs training data to build the model. We can use the existing cluster membership data or use other methods to first create the train data before running it into a random forest to detect more members from the new observations.

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

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

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

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.

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