Feasibility of Discovering Star Clusters in Gaia Data Using Unsupervised Clustering Algorithms

Dan Feldman

danfeldman90@gmail.com

Stellar clusters are important for determining ages and physical properties for their members, providing insights into star formation and evolution. Discovery of stellar clusters, especially at intermediate and large distances can be extremely difficult, as determining distances and velocities require precise measurements. The Gaia DR2 dataset contains the relevant data for upwards of 1.3 billion stars in the Milky Way Galaxy and is the most sensitive and expansive dataset of its kind to date. I found a number of interesting stellar sources, such as giants and hypervelocity stars, warranting further study. I used a subset of 144,806 sources in order to determine the ability of KMeans and DBSCAN clustering algorithms to identify realistic clusters within this dataset. KMeans required a two-step clustering process, though clusters identified were still too large to be realistic. DBSCAN performed better, though there were potential issues spatially and with

regards to artificially created boundaries. In the future, KMeans could be improved by identifying outliers, while utilizing the full DR2 dataset would truly test the ability of DBSCAN to find realistic clusters.

Received _____; accepted _____

1. Introduction

Stars are the engines that create the elements necessary for life, not only on Earth, but potentially elsewhere in the Universe. Our Sun provides us with heat and light, and without stars we could not exist. Understanding star formation and evolution is therefore an important endeavor in pursuing our own origins and our future relationship with the Sun.

The difficulty in studying stars is their long lifetimes; stars like our Sun are born over the course of a few million years, and live for around ten billion years. Given this reality, we would need to study stars like our Sun at different ages in order to understand its past and future. But aging individual stars is very difficult, as they spend most of their life as "main-sequence" stars, which keep similar properties for billions of years.

To get around this issue, astronomers can study stellar clusters; star clusters are thought to form together, making them all the same age. As a result, aging a cluster can be easier to accomplish, allowing for age determination for the individual stars inside them. Clusters come in two basic forms: open clusters and globular clusters. Open clusters tend to be smaller, younger, and more dispersed, staying bound for up to about 100 million years. Globular clusters are larger, denser, and longer-lived, as they tend not to disperse at all, staying bound for billions of years. Open clusters

are typically only found the the disk of our galaxy, while globular clusters can be in the disk, center, and halo of the galaxy. By finding clusters and identifying their members, we can develop a dataset of stars of different ages for evolutionary studies, as well as a variety of other interesting astronomical pursuits.

To identify stellar clusters, scientists must find stars that are both near each other spatially, and moving together through the galaxy. However, determining a star's distance from Earth is extremely challenging, as is determining its velocity; this is due to the large distances between us and the stars. Only recently have datasets been released that allow for such sensitive measurements to be obtained.

1.1. The Gaia Dataset

The Gaia satellite, launched by the European Space Agency (ESA) in December of 2013, is located at the Earth-Sun's L2 Lagrange point, where is in a stable orbit on the far side of the Earth from the Sun[1]. It uses a light shield to help reduce noise from the Sun, Earth, and Moon, and has been operating since. Its goal is to map the 3-D positions of over a billion stars in our galaxy, as well as calculate their movements and some stellar properties

[1]https://www.cosmos.esa.int/web/gaia/launch

for the sources (Gaia Collaboration et al. 2016).

The Gaia Data Release 2 (DR2) was released to the public on April 25th, 2018, and is the largest and most precise dataset of its type to date (Gaia Collaboration et al. 2018). It contains the 3-D positions of over 1.3 billion sources, as well as their 2-D tangential velocities (in Right Ascension and Declination). This makes it the perfect resource for discovering new stellar clusters, as well as discovering new members of previously identified clusters.

1.2. Potential Clients

There are a number of scientists currently working on studying the Gaia data for insights into stellar clusters. If I can utilize the unsupervised learning techniques commonly used in data science, then this could be very useful to those who are currently undergoing this work of cluster discovery and membership population. Many astronomers in recent years have made the switch to using Python as their main coding language, so utilizing Python packages for this work will make it easy for these scientists to utilize methods in this project. This work can be a branching point from which better techniques can be refined and utilized. As such, I will be exploring two clustering algorithms: KMeans and DBSCAN.

2. Data Acquisition and Cleaning

In order to acquire the Gaia data for my dataset, I needed to submit relevant queries to the Gaia@AIP web form[2]. The interface takes queries using either PostgreSQL or ADQL (Astronomical Data Query Language). While queries can be made via a guest account, I set up my own account in order to make queries that could take longer execution time. The data was acquired from the Gaia Data Release 2 Source Catalog (DR2; Gaia Collaboration et al. (2018)), which was released to the public in 2018.

The data selection for my dataset was four million sources, grabbed in two nearby sections of the source catalog. Ordering by source_id, I queried for two million sources with an offset of ten thousand, and then for another two million with an offset of three million. In this way, I created a subset that was nearby but not continuous in space. From these sources I queried for the following fields in the database: source_id, right ascension (RA), ra_error, declination (Dec), dec_error, parallax, parallax_error, Proper Motion in RA (pmra), pmra_error, Proper Motion in Dec (pmdec), pmdec_error, duplicated_source (boolean), Radial Velocity, radial_velocity_error, Effective Temperature (teff_val), and Luminosity (lum_val). The data files downloaded containing these data are too large to be hosted on the GitHub repository

[2]https://gaia.aip.de/query/

for this project, but can be easily downloaded using the aforementioned webform.

The most important features needed for the study were the three spatial dimensions (RA, Dec, and Distance), and the two velocity dimensions in the RA and Dec dimensions. Radial Velocity would also be an important velocity vector, but too few sources in the dataset had that velocity data available. A complicating factor to obtaining the distance is in the proper conversion between the parallax angle measured by Gaia and the distance in parsecs. Normally, this is an easy conversion defined by the equation:

$$\mathrm{d} = \frac{1}{\theta} \tag{1}$$

where θ is the parallax angle in arcseconds. However, Equation 1 only works when the error in the parallax is sufficiently small, but this is not the case for many of the sources in the DR2 source catalog. As such, a method for calculating distances was detailed and calculated in Bailer-Jones et al. (2018). In order to access the distances calculated in Bailer-Jones et al. (2018), I needed to query the ESA Gaia Archive [3] for the relevant data. The important fields to the study were the estimated distance (r_est), and the upper and lower distance limits (r_hi, r_lo respectively).The files containing

[3]http://gea.esac.esa.int/archive/

these data were too large to host in the GitHub repository.

Given the constraints of my computer, I needed to further cut my data into a smaller subset. As such, I grabbed the first 100,000 sources from each of the two million subsets I had downloaded from the source catalog, starting out with a total dataset of 200,000 sources. From there, I needed to clean up the data. First, I cut out any of the sources that did not have a physical parallax value. Then, I removed all sources that are considered duplicates by cutting any with the duplicate flag. At this stage, all of the data had the necessary five dimensional data I would be using to conduct the study, so no further cuts were necessary. In the end, my dataset contained a total of 144,806 sources. I saved this file in a pickle file called "cleaned_dataframe_subset.pkl" and stored it in the Data folder of my GitHub repository.

3. Exploratory Data Analysis (EDA) and Inferential Statistics

Exploratory Data Analysis is a crucial piece of any data science project, shedding light on any insights into our dataset before any kind of modeling is carried out. These insights can often have important bearings on the analysis or results that come later, and can also illuminate trends relevant to the problem being looked at. For this project, I looked into the distributions

present in the data, how they are related, and if any trends are apparent.

3.1. The Five-Dimensional Distributions

For my dataset, I was going to be looking for stellar clusters among 144,806 sources, so it was important to look into what the distributions look like in all five relevant dimensions for our sample. The first I looked at were the three spatial distributions of the sample, which are the RA, Dec, and Parallactic Distance (r). The RA and Dec are given in units of degrees, and the Parallactic Distance is given in parsecs, which is an astronomical unit of distance equal to 3.26 light years, or 3.086 x 10^{16} meters. These data are plotted in Figure 1.

Figure 2 shows the distribution of parallactic distances for the sample. Farther stars are both dimmer and have smaller parallax angles, so the falloff at larger distances is likely due to a selection bias. Despite that, there are still a number of sources at appreciable distances from Earth.

Next, I explored the velocity distributions for my sample. Given the dearth of radial velocities (velocity in the direction of our line-of-sight towards the source), I am only considering the velocities in the RA and Dec directions, which together form the tangential velocity. Figure 3 shows the proper motions for the sample, split into the two subsets, shown in blue and

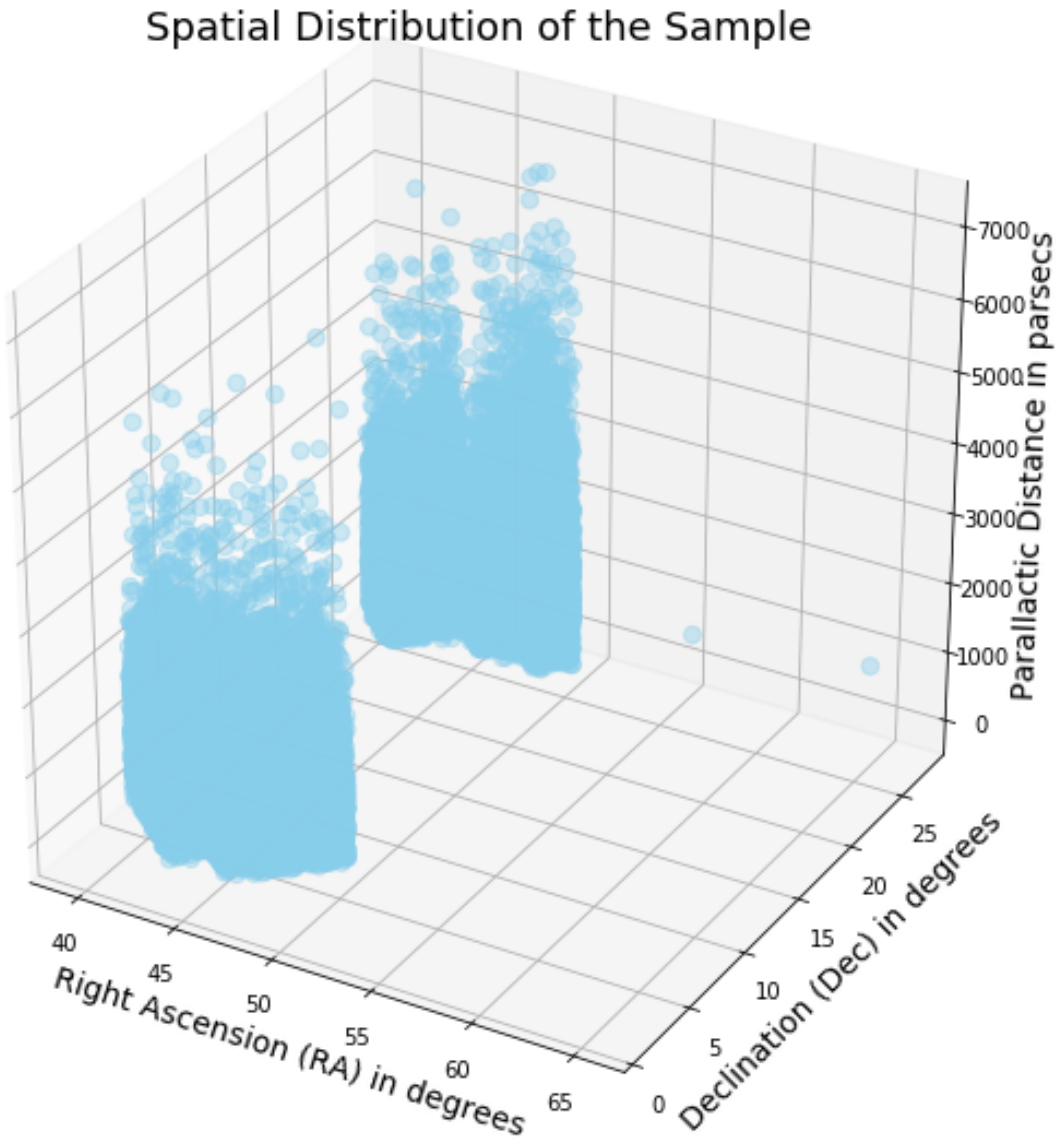

Fig. 1.— The three spatial dimension data for my dataset. It is clear from this plot that the sample is split into two distinct groups of sources, and the sources are denser at closer distances to Earth. For reference, the galactic center is at approximately 8000 parsecs from Earth.

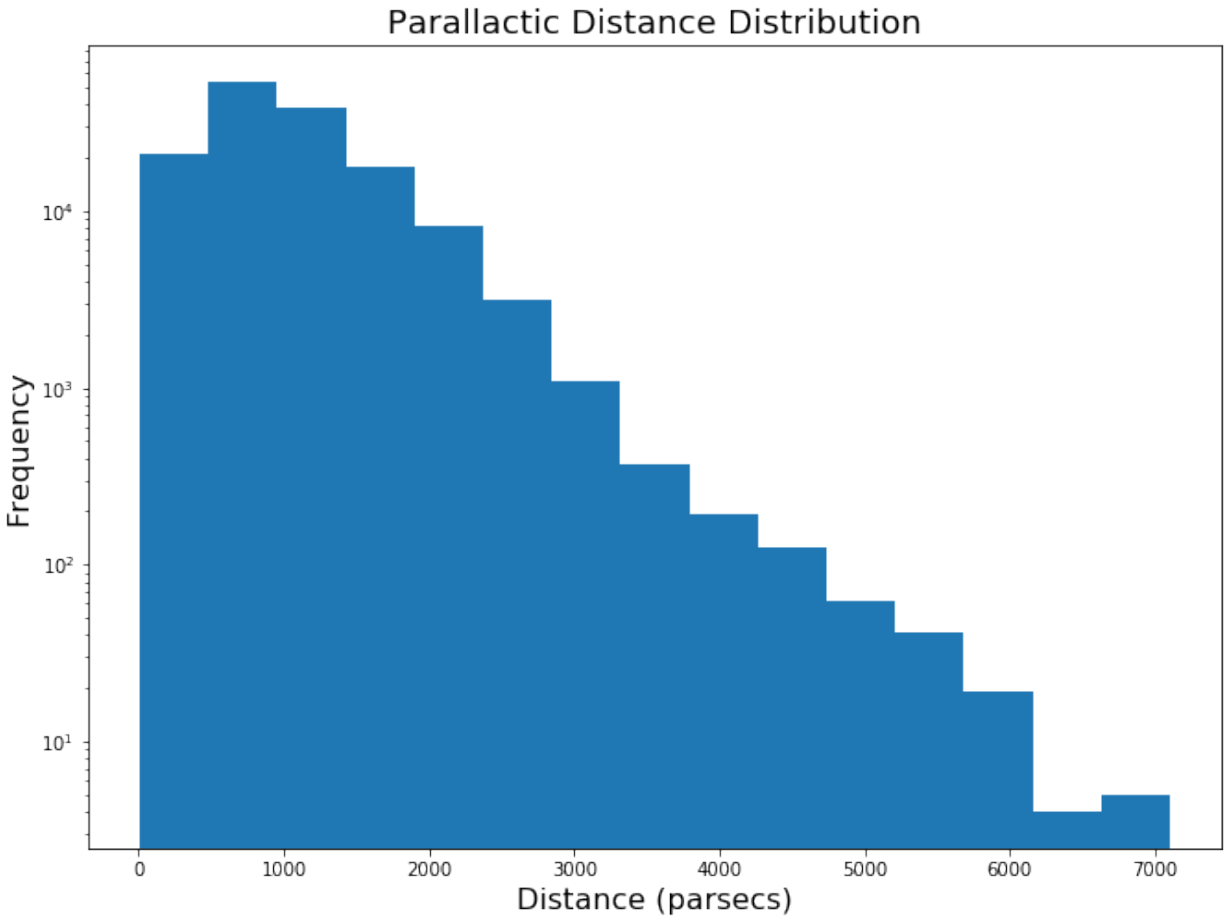

Fig. 2.— The parallactic distance distribution for my sample. It is shown how most of the sources are located closer to Earth, though there are still a number of them at larger distance. This is likely in part a selection bias, as it is harder to measure parallax for more distant stars, both because the parallax is smaller in angle and because the stars are dimmer.

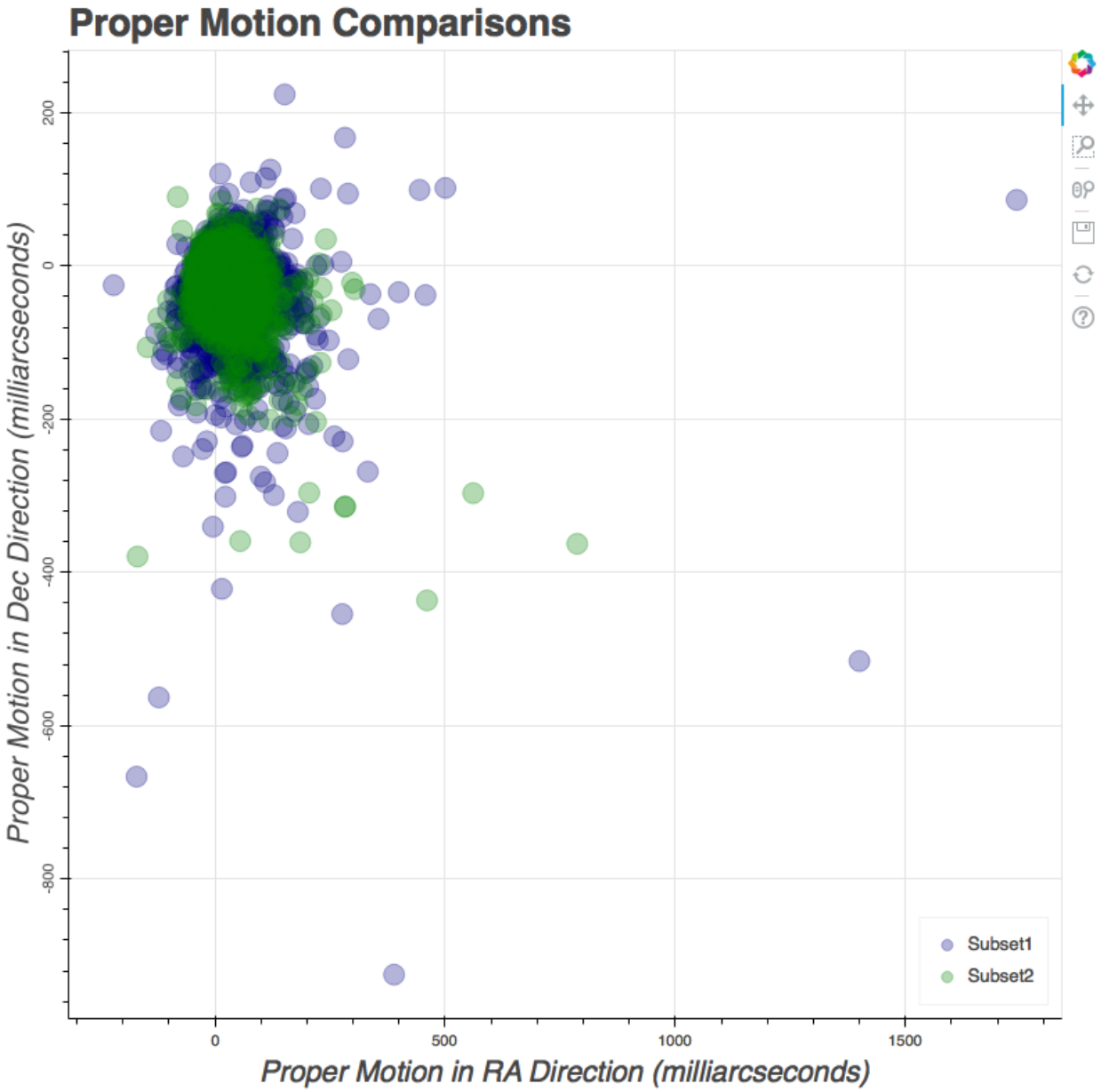

Fig. 3.— The comparison in the proper motions for the two subsets of my sample, shown in green and blue. The plot seems to make clear there is no major difference between the two subsets.

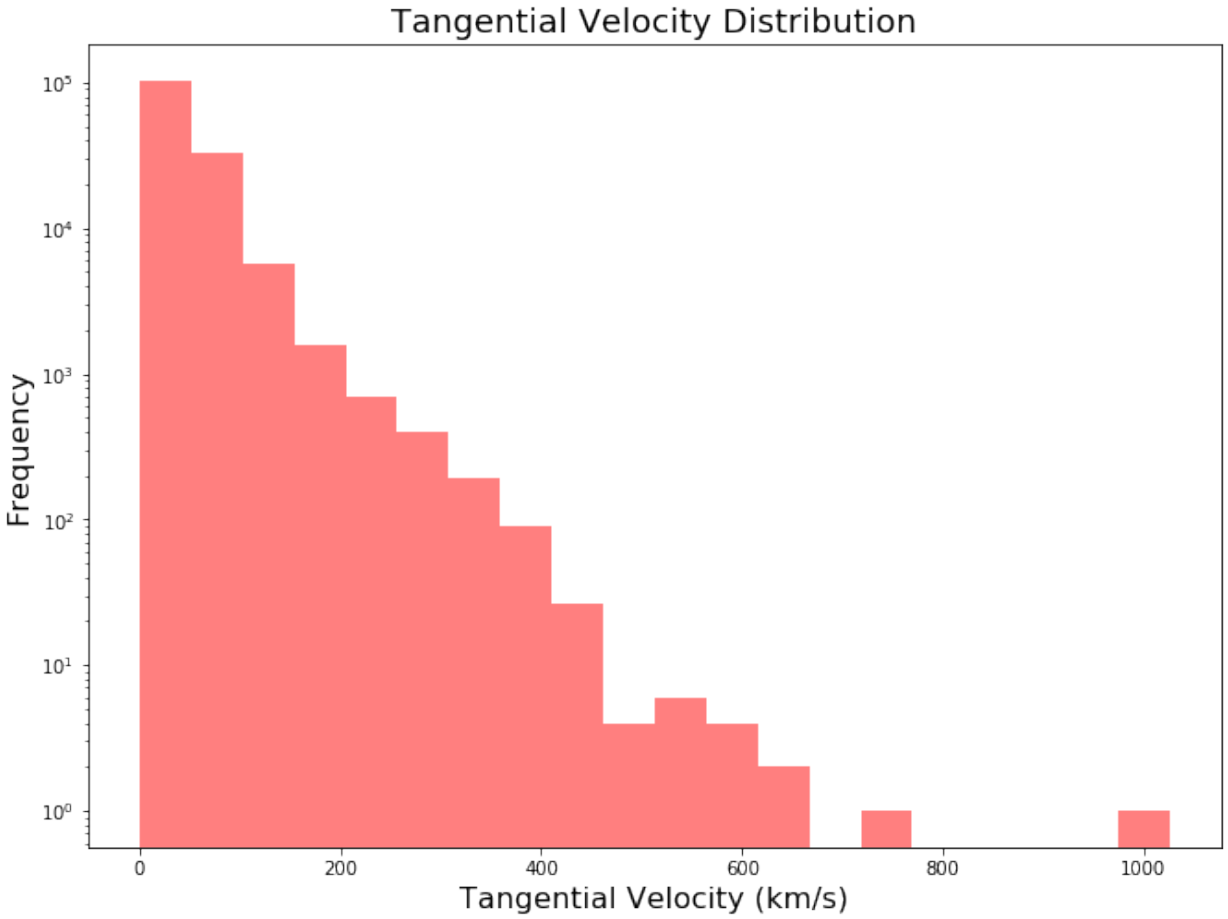

Fig. 4.— The tangential velocity distribution for my sample, given in kilometers per second. The tangential velocity is the total velocity for the sources in the directions tangential to our line-of-sight towards the source. All sources with tangential velocities greater than 550 km/s are likely escaping the Milky Way Galaxy, and the two fastest sources are fast enough to be considered hyper-velocity sources.

green. No appreciable difference can be seen between the two subsets.

I calculated the tangential velocity distribution by taking the square root of the squares of each component in the tangential velocity, which are plotted in Figure 4. While most of the sources are moving at speeds typical for stars in the galaxy, there are a number of sources moving very fast. Stars moving faster than 400 km/s are high-velocity sources, that in many cases are likely in the process of escaping the gravitational well of the Milky Way Galaxy and will fly out. The escape velocity of the Milky Way is roughly 550 km/s, to give a basis for speed, and this velocity is missing the radial component, which when added would increase the total value for escaping.

Some stars in the last decade or so have been discovered to have velocities that are extremely high compared to the typical galactic population; these stars have been called hyper-velocity stars[4]. While different sources define these stars with different minimum velocities, the low end ranges from 700 to 1000 km/s. As such, there are one or two stars in the sample that have enough speed to be considered hyper-velocity stars with their tangential velocity alone. These stars likely gained such high velocities due to interactions with the supermassive black hole in the center of our galaxy (Sag A*)[5], and would be interesting to study in their own right.

[4]https://physicstoday.scitation.org/doi/10.1063/PT.3.3199

[5]https://www.nasa.gov/mission_pages/chandra/multimedia/black-hole-SagittariusA.html

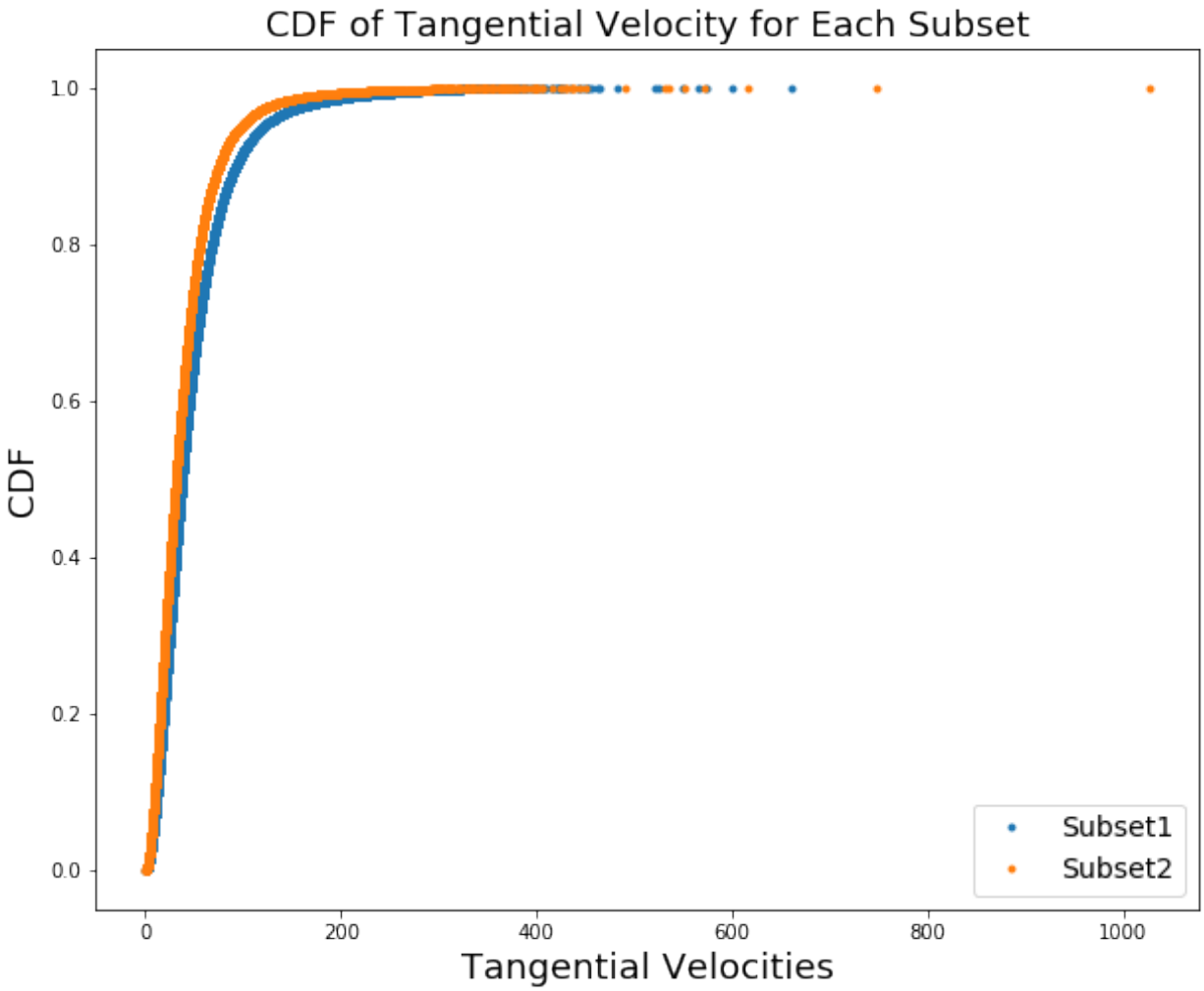

Fig. 5.— The Cumulative Distribution Function (CDF) for each of the subsets in my sample.

Another interesting thing to look at whether or not the tangential velocity distributions differ at all between the two subsets in my sample. These distributions are plotted as Cumulative Distribution Functions (CDFs) in Figure 5. The two distributions look similar, but there could be a statistically significant distance. In order to better characterize these differences, I can model each distribution as a log-normal distribution. To check if this is a fair comparison, I plotted the tangential velocity distribution against a log-normal distribution of similar μ and σ, as well as the log-distribution of the tangential velocity against a normal distribution (since the log of a log-normal distribution is a normal distribution, hence the name). These are shown in Figures 6 and 7, respectively. The comparisons seem very good, and so I continued to characterize the tangential velocity distributions as log-normal distributions.

I then ran a z-test in order to determine whether or not the two subsets are log-normal distributions with statistically significantly different means and standard deviations using both the combination of μ and σ and arithmetic mean and standard deviation. In both cases, I was able to show with very high confidence that the two distributions are not the same, with p-values of 0. This result seems to suggest that the tangential velocity distribution varies as you look at different areas of sky; this is not inherently surprising, though given the relative closeness of the subsets, I

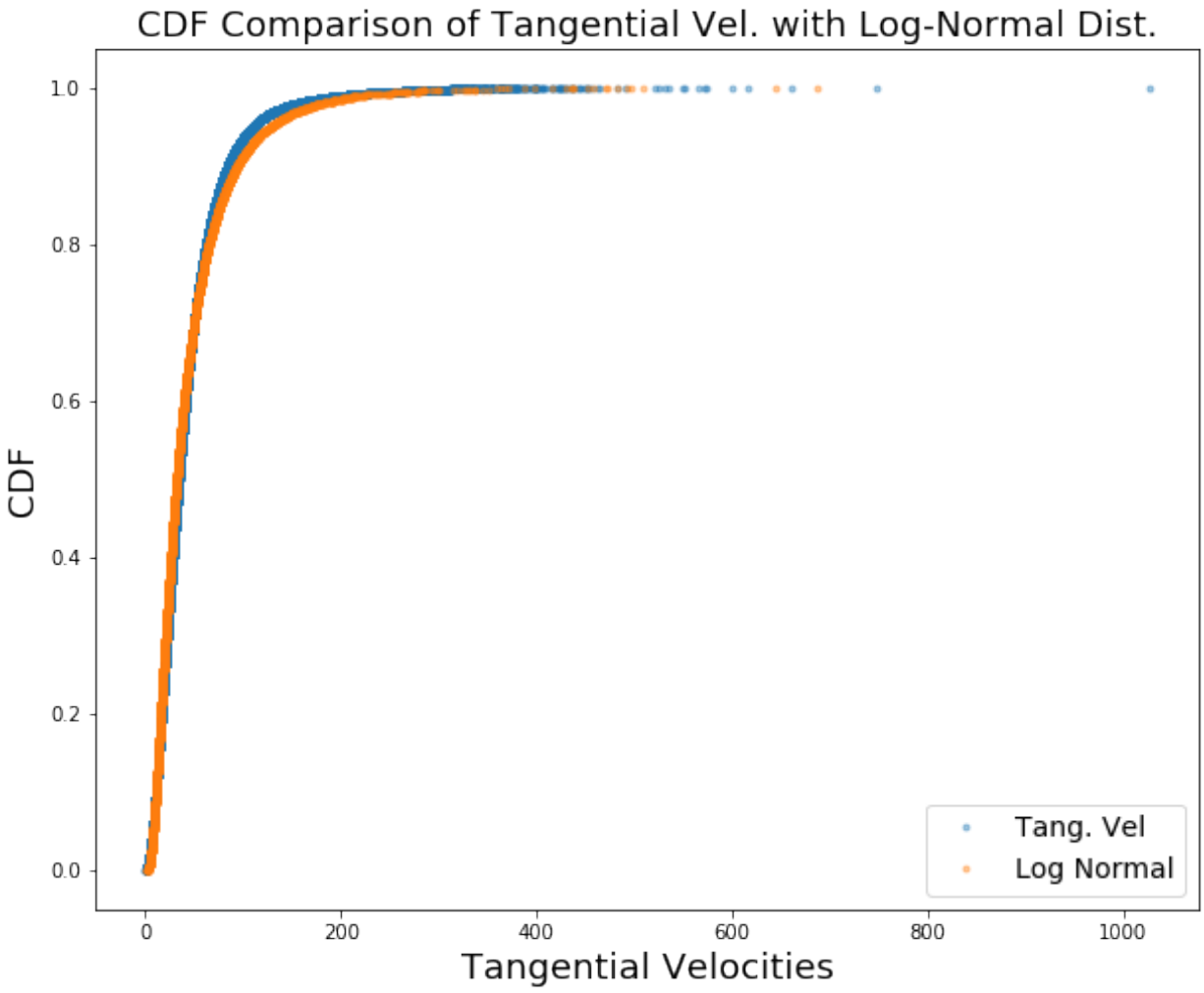

Fig. 6.— The Cumulative Distribution Function (CDF) for the tangential velocity of the sample plotted against the CDF of a log-normal distribution with the same parameters.

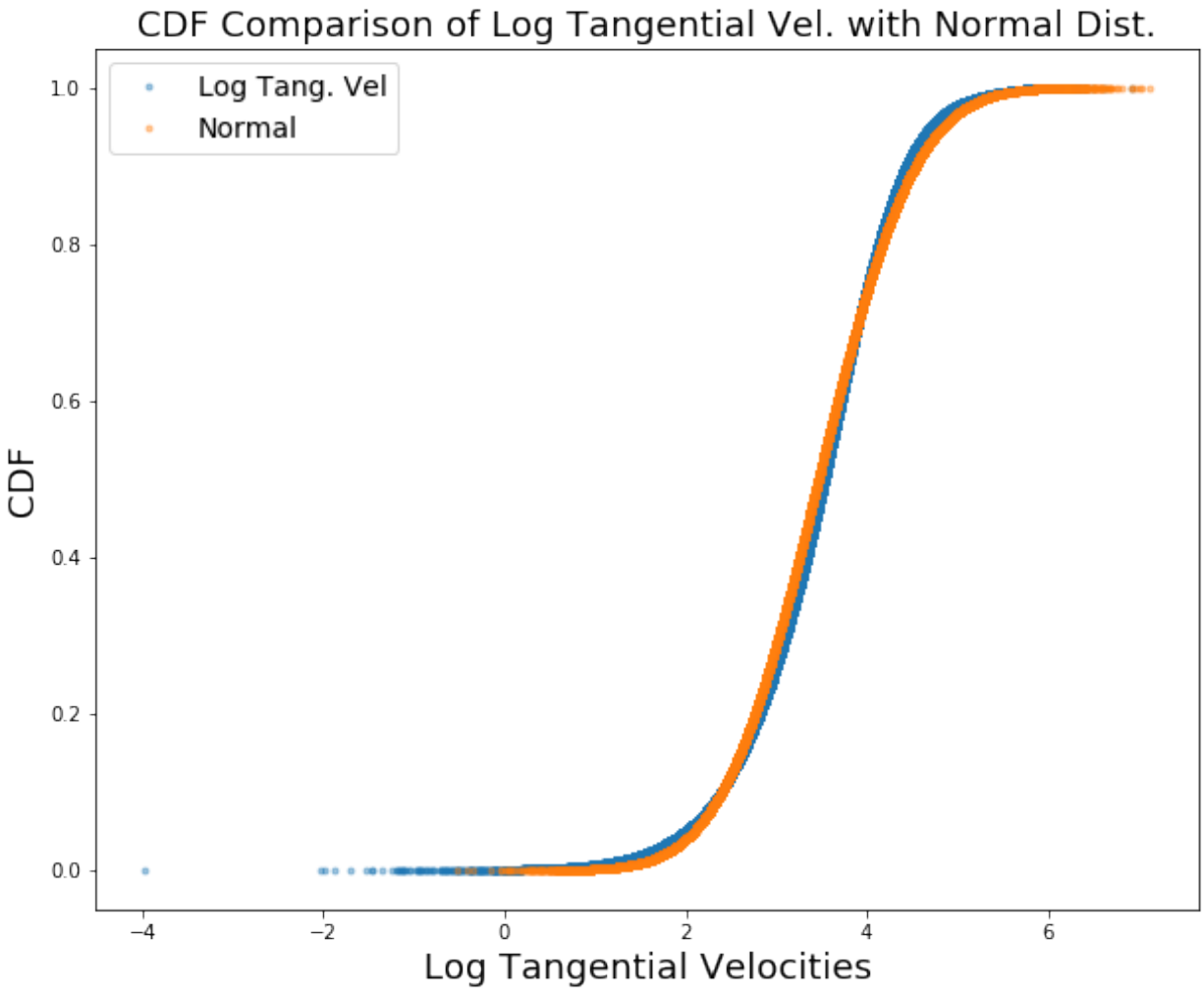

Fig. 7.— The Cumulative Distribution Function (CDF) for the log tangential velocity of the sample plotted against the CDF of a normal distribution with the same parameters.

am struck that this result is still recovered. I had expected that they'd be less distinguishable than they ended up being.

Lastly, I calculated the pearson correlation coefficient between the tangential velocity for my sample and the distance, to see if there is a correlation between the two. I calculated a value of 0.12 and a p-value of 0.0, suggesting a very slight positive correlation. This is interesting, as you would expect the opposite result. Observational data has shown that stars near the galactic center have much lower velocities than that of the stars further out, which oscillate a bit but more or less level off after 3,000 parsecs from the galactic center. Given that the farthest stars in my sample are greater than 3,000 parsecs from the center, it makes more sense that any correlation found would be close to zero, though if any correlation would be found, it would still be more likely to be slightly negative. This could also be explained in part by selection biases in the sample.

3.2. Stellar Parameters

While most of the sources in my sample did not have stellar parameters attached to them, 27,605 sources contained information for their luminosities and effective temperatures, allowing for study of their collective properties. To begin, I calculated the pearson correlation coefficients between three

properties of the stars: luminosity, effective temperature, and distance. The results are shown in the second column of Table 1, under "All Stars."

Things became more interesting when I plotted a Hertzsprung-Russell (H-R) Diagram of the stars, plotting their effective temperatures on the x-axis and the luminosities on the y-axis, shown in Figure 8. This illustrated that a large number of the stars (882 in total) in my sample are giants (and supergiants), shown in red. As such, their properties are potentially going to be different than main-sequence stars, shown in blue. I subsequently created scatter plots for the remaining features, keeping the same color scheme, shown in Figures 9 and 10. In both of these plots, I look at distance on the x-axis and a stellar property on the y-axis. In both cases, main-sequence stars have a cone shape in the plot. The reason for this is that smaller stars will be more difficult to measure at large distances, so the lower-right corner of the plot is unpopulated, indicative of a selection bias in the data. While it is true that almost all of the closest stars to Earth are small and cool in temperature, there is no reason to expect a dearth of said stars at larger distances. Giants, however, are easy to detect at all distances in our sample, and appear as flat lines in both plots, as they are all very luminous, and very cool in temperature. It makes astronomical sense for distance to play little-to-no role in the stellar properties of stars, as they are determined by the processes internal to the star.

Having separated out the giants and main-sequence stars, I re-calculated their pearson correlation coefficients, which are listed in the third and fourth columns of Table 1. These values reflect well what is demonstrated in the data visualizations.

Lastly, I wished to determine if the giants have a different tangential velocity distribution than their main-sequence counterparts. As the speeds are not determined via internal properties, but external ones, it is interesting to examine if such a difference exists. I plotted the effective temperature versus tangential velocity for each of the stars in the same, using the same color scheme for giants and main-sequence stars, shown in Figure 11. I also plotted the CDFs for the main-sequence and giant tangential velocities, shown in Figure 12. The CDF betrays what appears to be a clear difference in distribution, where giants are more heavily skewed to the higher velocities.

I ran a z-test for these two distributions, assuming log-normality, and verified that these two distributions have different properties, with a p-value of 0.0001. I am unsure whether this finding is indicative of a further selection bias in our sample, or of an astronomical phenomenon. It is certainly worth further study.

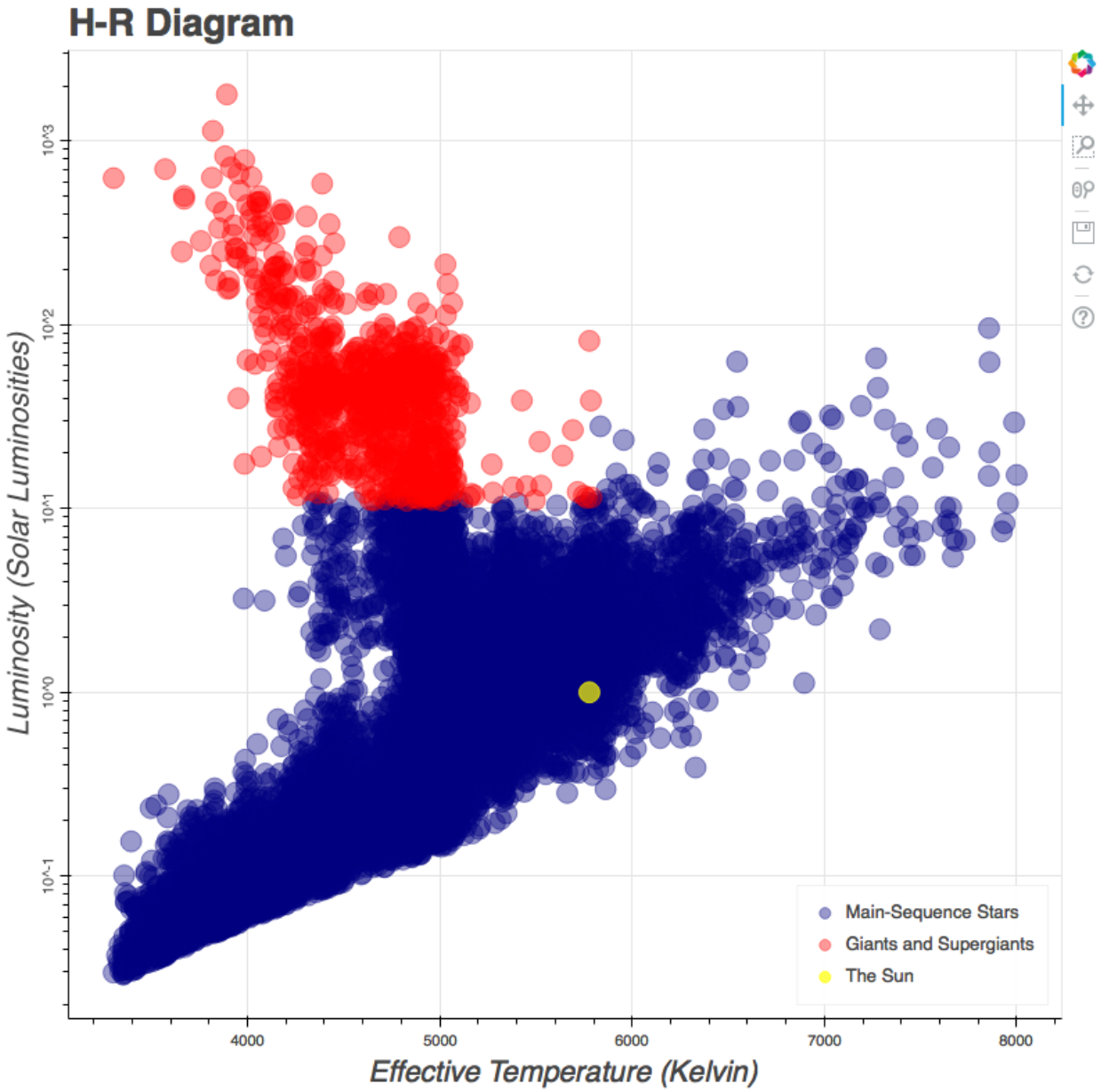

Fig. 8.— An H-R Diagram for the sources in the sample containing stellar parameters. In blue, we see stars along the main sequence, while in red, we see the giants and supergiants. I have placed a yellow dot on the plot to indicate where the Sun would be if it were a part of the sample. The cut made to differentiate between giants and main-sequence stars was arbitrarily made to be stars with a luminosity greater than 11 solar luminosities and effective temperatures less than 5800 Kelvin.

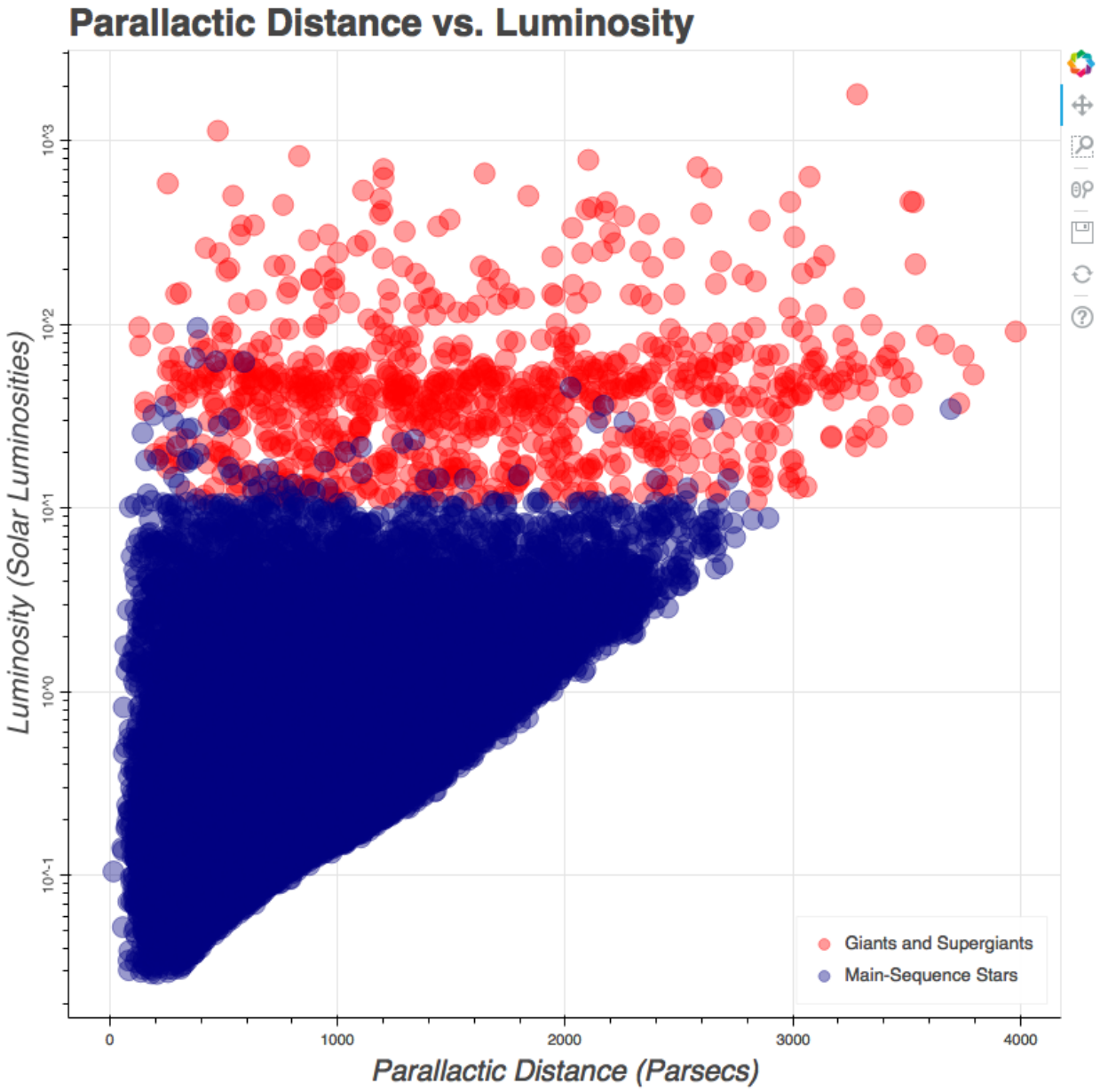

Fig. 9.— A scatter plot showing the distance versus luminosity for the stars in my sample. Main-sequence stars are shown in blue, while giants are shown in red. The main-sequence stars appear to have a cone-like shape, illustrating the selection bias in the sample, while giants appear as a flat line, as expected.

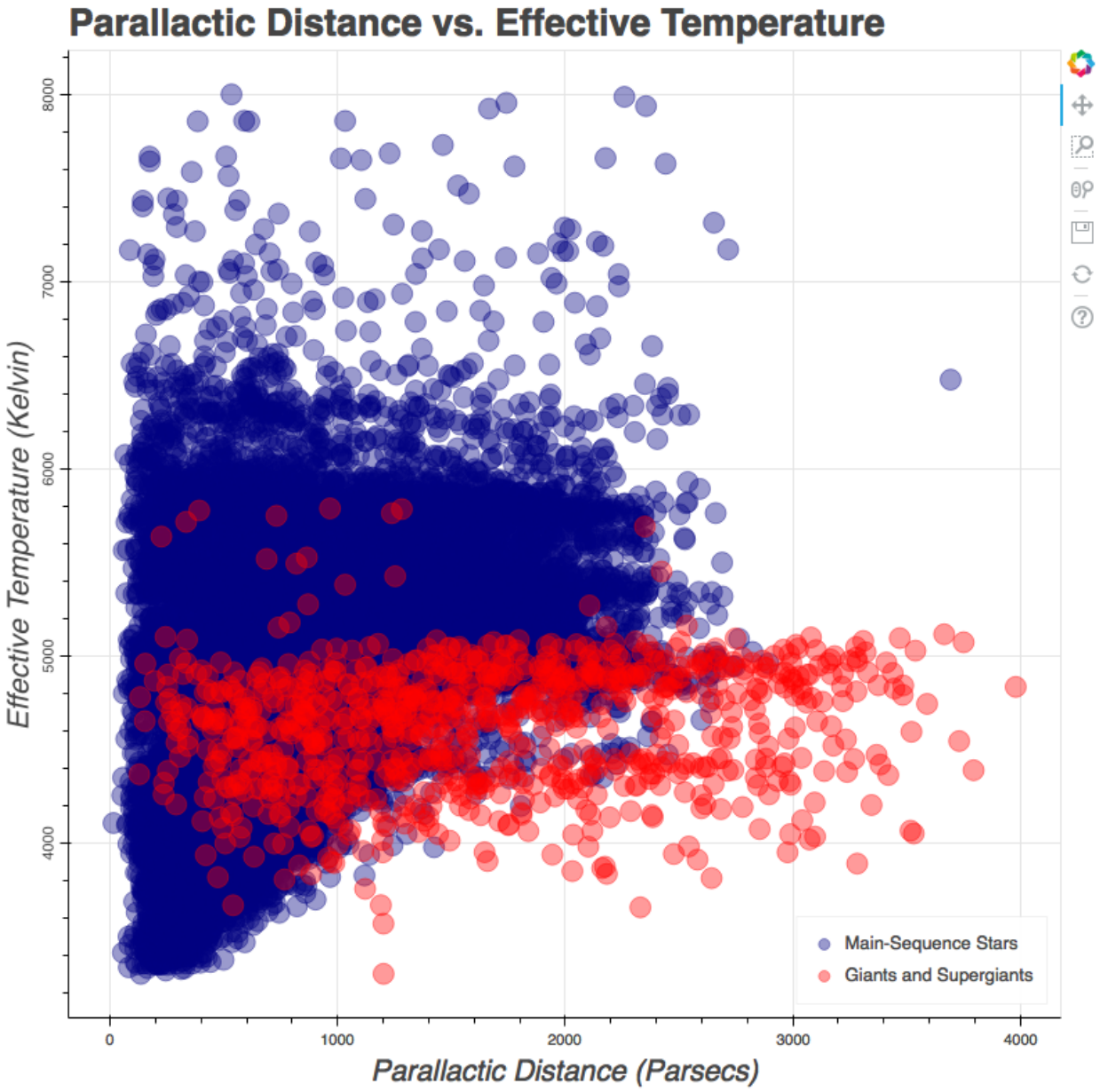

Fig. 10.— A scatter plot showing the distance versus effective temperature for the stars in my sample. Main-sequence stars are shown in blue, while giants are shown in red. The main-sequence stars appear to have a cone-like shape, illustrating the selection bias in the sample, while giants appear as a flat line, as expected.

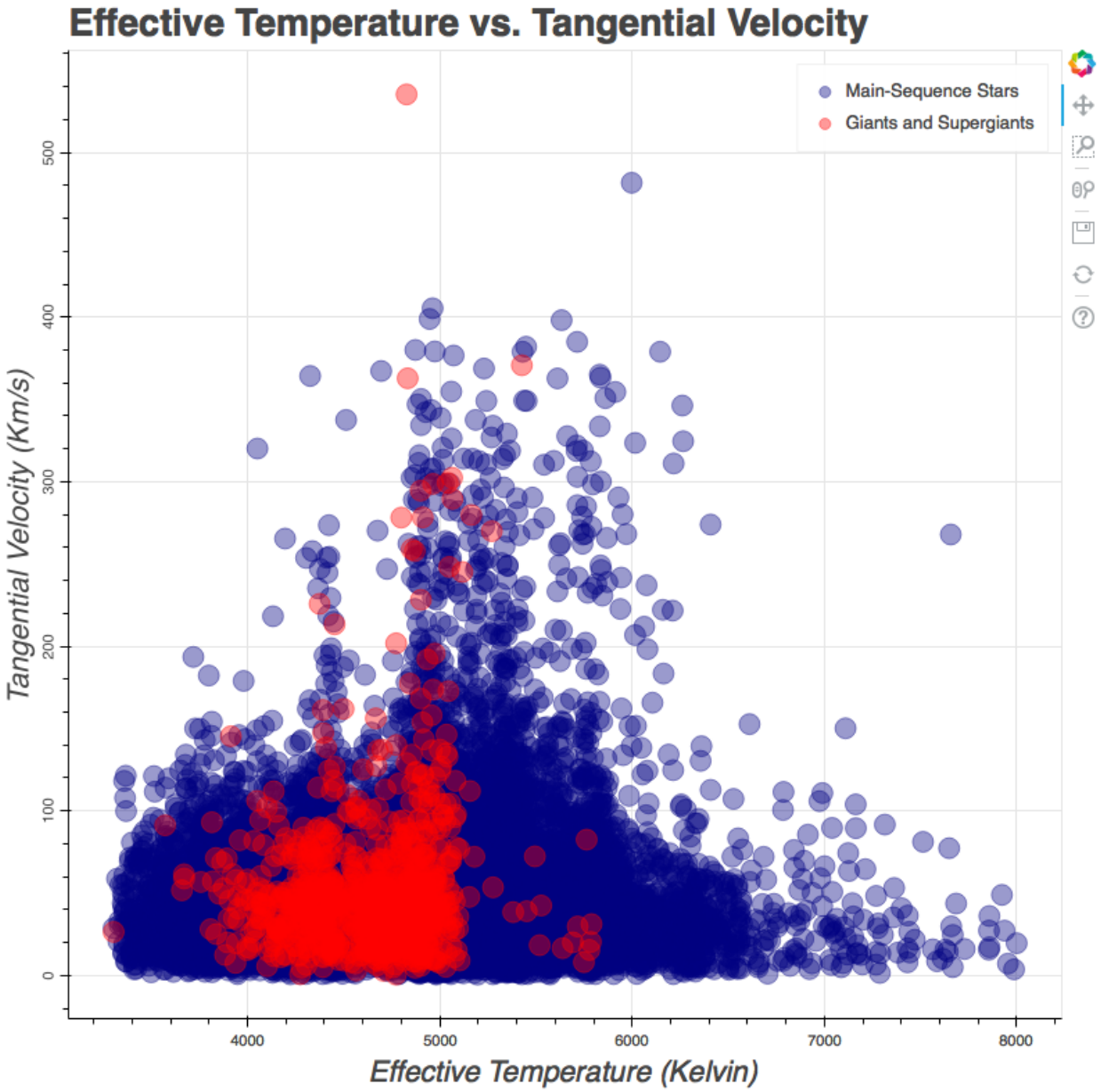

Fig. 11.— An H-R Diagram for the sources in the sample containing stellar parameters. In blue, we see stars along the main sequence, while in red, we see the giants and supergiants. I have placed a yellow dot on the plot to indicate where the Sun would be if it were a part of the sample. The cut made to differentiate between giants and main-sequence stars was arbitrarily made to be stars with a luminosity greater than 11 solar luminosities and effective temperatures less than 5800 Kelvin.

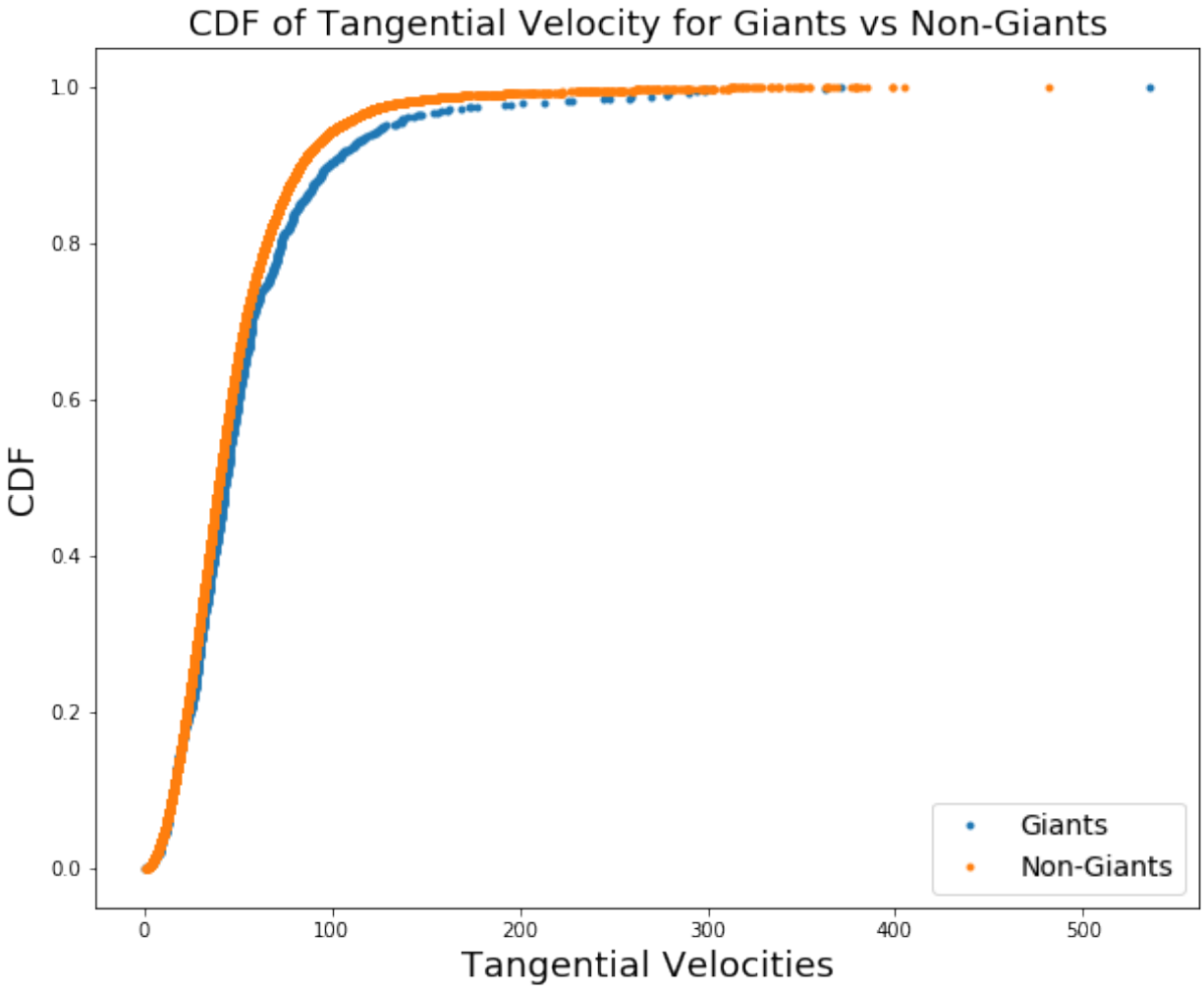

Fig. 12.— The Cumulative Distribution Function (CDF) for the tangential velocity of the main-sequence stars plotted against the same CDF for the giant stars. A difference in distribution seems clearly evident.

4. Analysis and Results

The core question behind this project is whether or not unsupervised learning clustering algorithms can translate well into finding clusters of stars in Gaia data. There are a number of algorithms to try, but I focused on two: KMeans and DBSCAN. In the case of KMeans, the number of clusters (k) needs to be specified as a part of the algorithm, while DBSCAN does not have that requirement. In the following sections, I'll discuss each of these algorithms, how I approached the modeling, and the results they each produced.

4.1. KMeans

For this type of problem, I have 5-dimensional, numeric features and wish to cluster the sources by a metric of distance between these variables; KMeans is a natural choice for such a problem, as it does just that. One key detail when dealing with the KMeans algorithm is figuring out how much weight to put on each of the variables. If I use the natural units the data came in, then the differences in the spatial dimensions are small in RA and Dec compared to the differences in the other three variables. As such, they will have little weight. By changing the units my data comes in, I can introduce a weighting to attempt to give them all a fair chance at being close

to equal. As such, I converted the RA and Dec into arcminutes, and my proper motions into kilometers per second, which isn't perfect, but puts the features at similar orders of magnitude, while still preserving their physical nature.

In order to come up with a cluster number, I used the elbow method, in which I ran the KMeans algorithm for my data using k values between 2 and 28 (in steps of 2) and scored the algorithm using sklearn's Kmeans.score() method, in which values closer to zero are best. I plotted the scores generated for each of these k values, and look for an elbow in the plot, shown in Figure 13. It was difficult to figure out where a clear elbow might exist, as you still get a fair amount of improvement despite the leveling off. I ultimately decided on 18 for the elbow.

Despite the changing of the units in my features, I still encountered a major problem – for the most part, only the spatial dimensions seemed to be considered for my clusters, such that the velocity dimensions were too spread out in the clusters. If I then weighted the velocity dimensions so that they order themselves, then the spatial dimensions became too spread out. This problem is illustrated in Figures 14 and 15, in which I used only consideration for the velocity vectors in the clustering algorithm. On the flip side, Figures 16 and 17 show the same plots but using all five dimensions. It was clear that in order to get all 5 of my features to be considered, I would

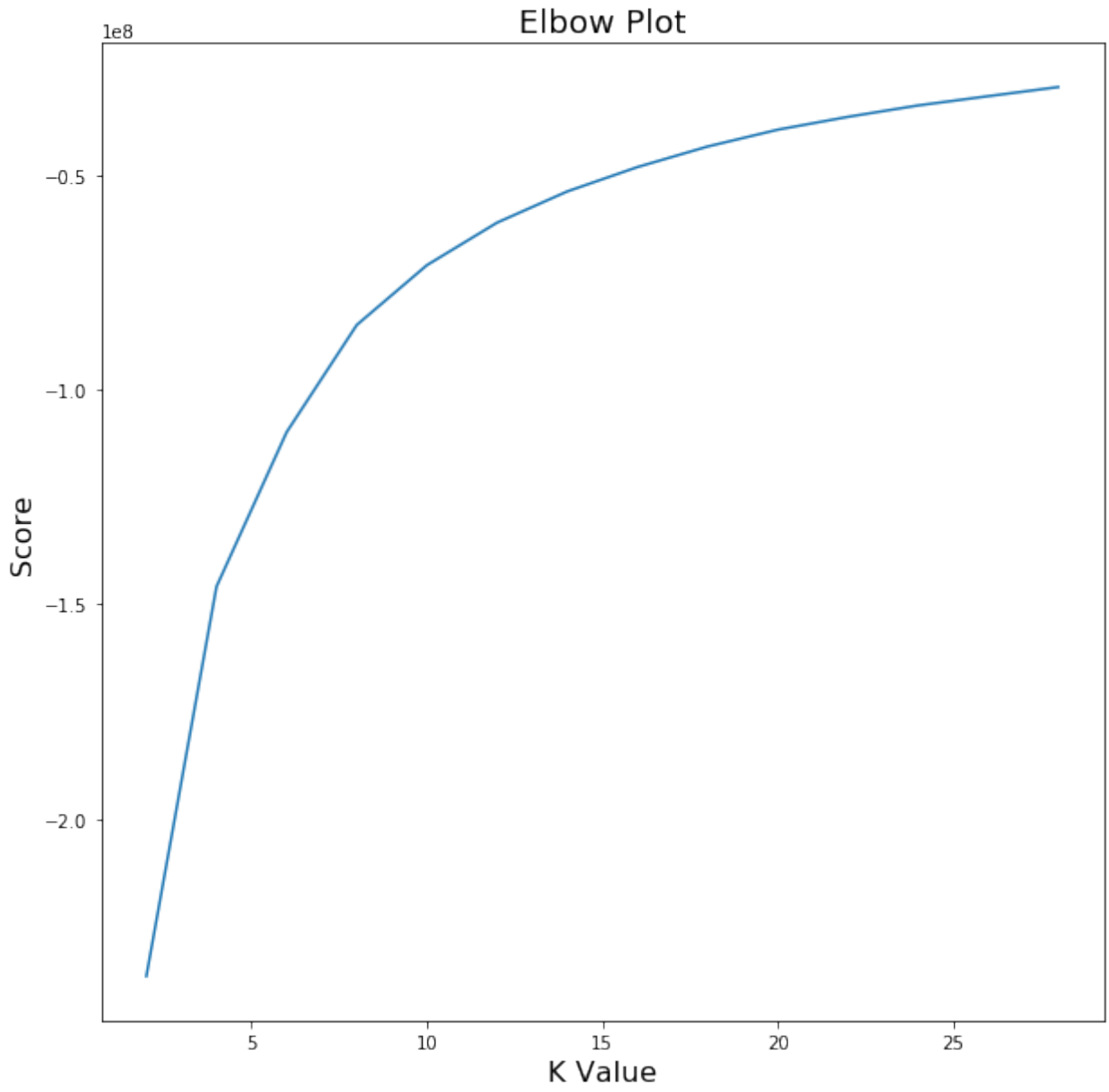

Fig. 13.— The elbow plot for my KMeans clustering, with values of k ranging from 2 to 28. No clear elbow was seen in my opinion; I ultimately chose 18 as my elbow.

need to alter my approach.

In order to address this problem, I decided to do a two-step approach to KMeans. First, I used the velocity features to split my dataset into the initial 18 clusters shown above in Figure 14. Then, I ran a second KMeans algorithm on each of the 18 clusters, this time using only the spatial data. For the second level of KMeans, I needed to automate the process of determining the number of clusters to use. I chose to keep using the elbow method, this time calculating the slopes between each two values of k and choosing a threshold in which the slope became flat enough to cut off and consider my elbow. For each subcluster, I tested k values between 1 and 18 (in steps of 1), and if the slope didn't get flat enough, the algorithm would choose 18, which did not happen for any of my clusters. Choosing a slope of less than 0.003, all of the k values chosen were between 14 and 17. Using this method, I ended with a total number of 281 clusters. The resulting clusters are plotted in Figures 18 and 19.

4.2. DBSCAN

The advantage DBSCAN has over KMeans is that you do not need to specify the number of clusters a priori. Instead, DBSCAN figures out not only how many clusters you have, but what data points do not belong to

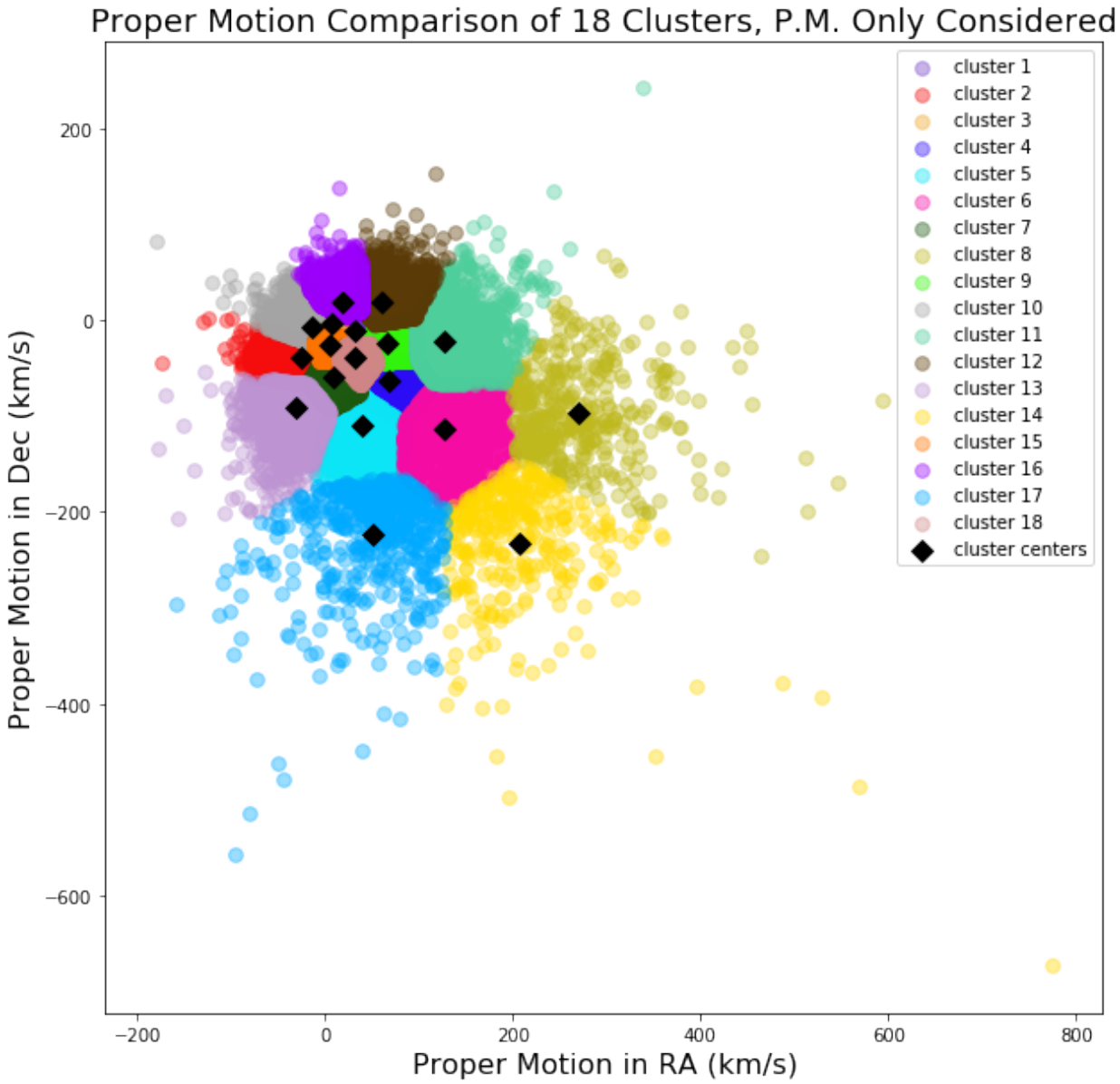

Fig. 14.— The clustering results in velocity space, colored by cluster membership. Cluster centers are shown in black diamonds. The clusters are clearly separated, as the separation was done taking only the velocity dimensions into account.

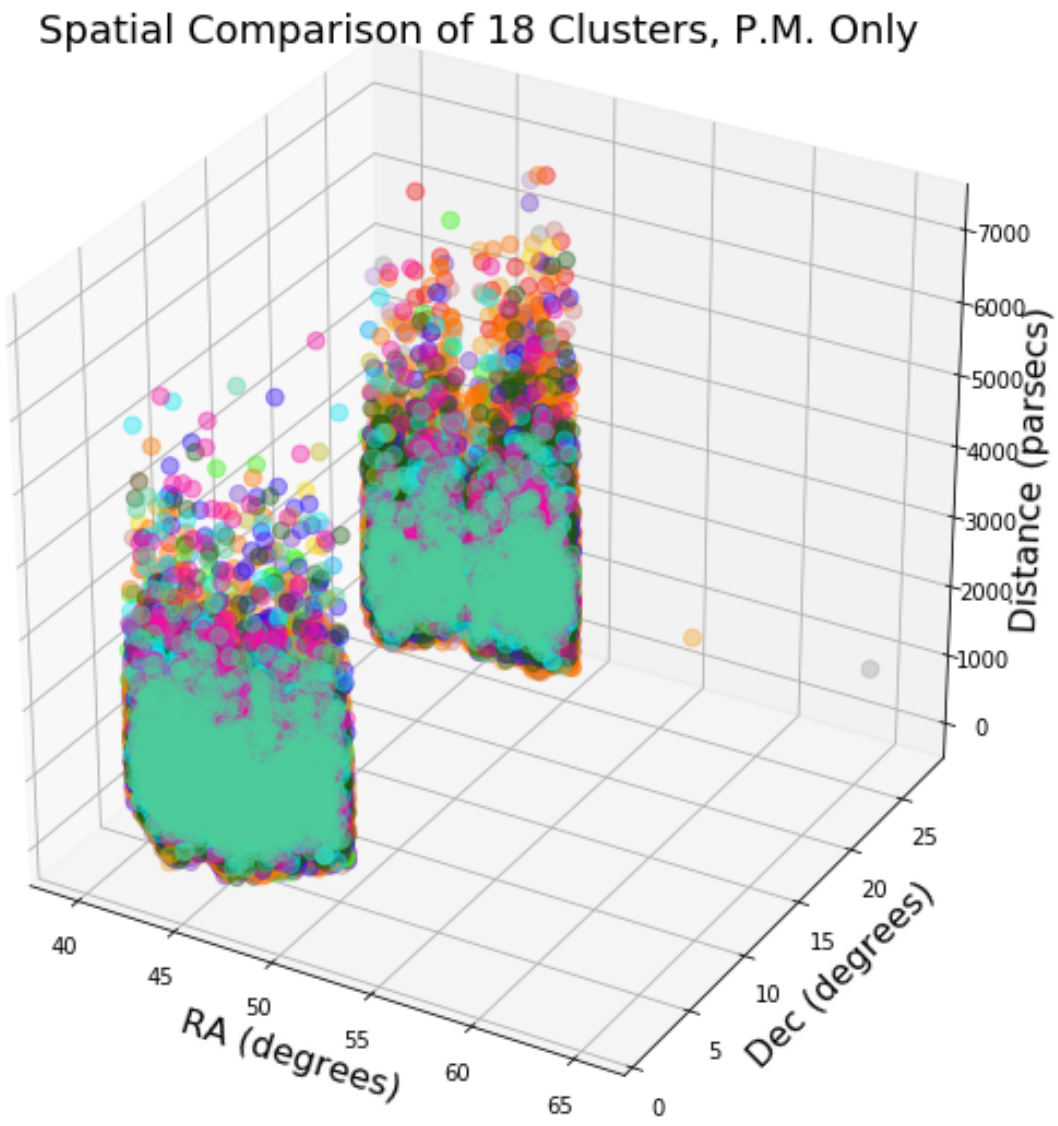

Fig. 15.— The clustering results in the three spatial dimensions, when only velocity dimensions were considered. There are 18 clusters, colored by membership in the plot. It is clear that when this is done, clusters are all over the place, and are not physical in their spatial dispersion.

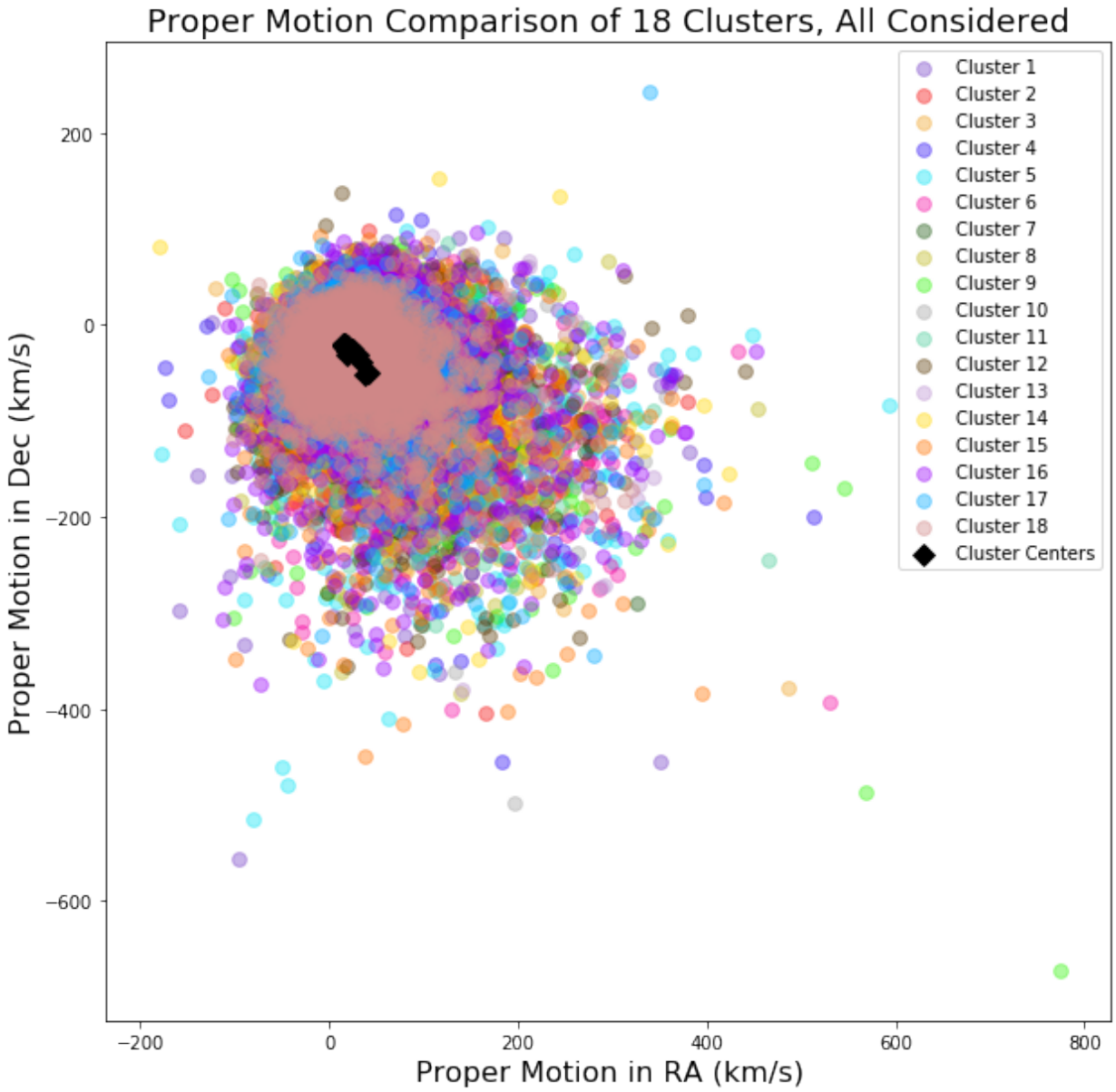

Fig. 16.— The clustering results in velocity space, colored by cluster membership. Cluster centers are shown in black diamonds. Given that all five dimensions were now considered, the velocity dispersions are much larger and unphysical.

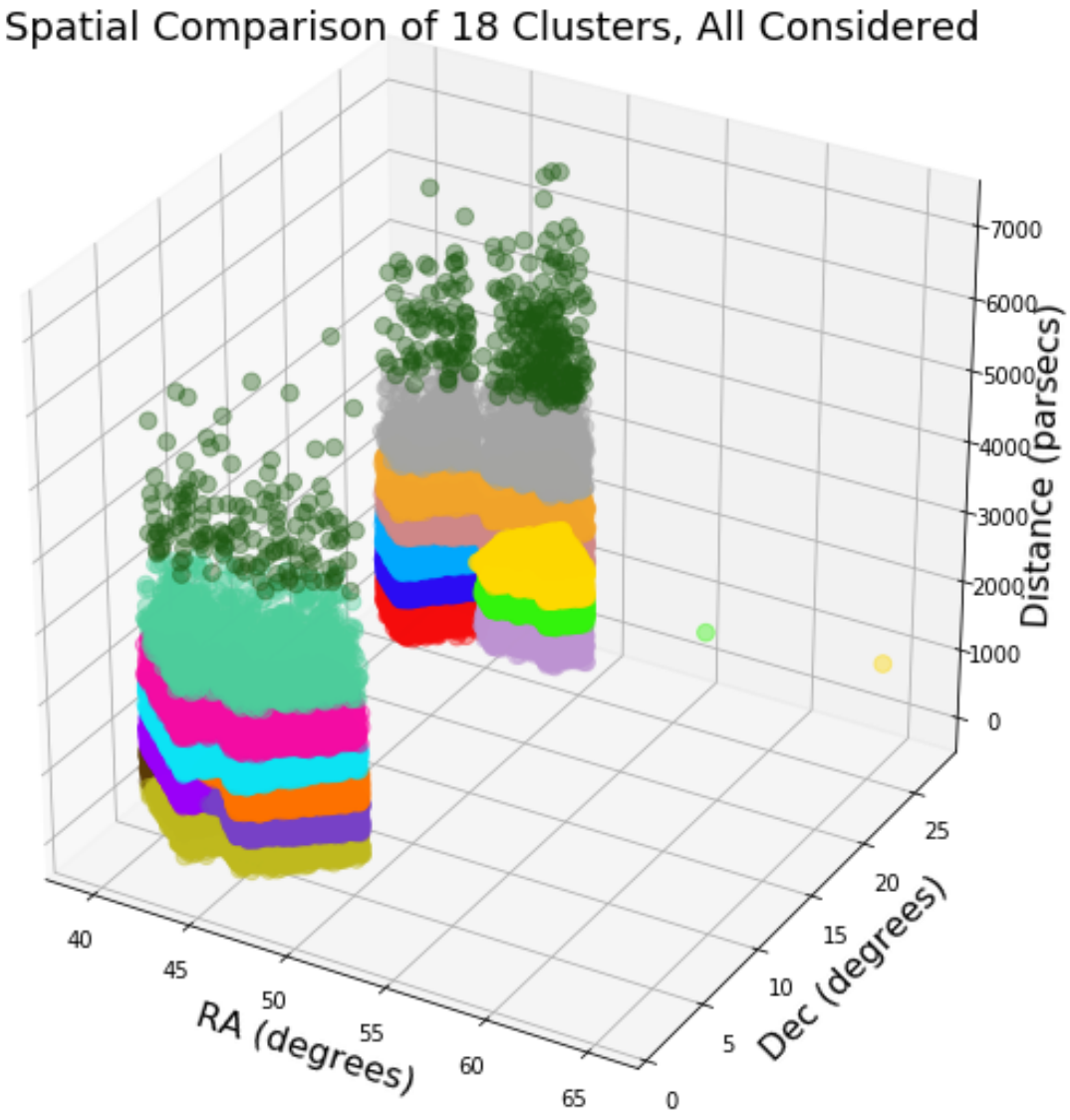

Fig. 17.— The clustering results in the three spatial dimensions, when all 5 dimensions were considered for clustering. There are 18 clusters, colored by membership in the plot. It is clear that when this is done, clusters are more ordered in space, and are more physical than the velocity dimensions. However, there is an issue with the cluster containing the farthest members being in both subsets, shown in dark green.

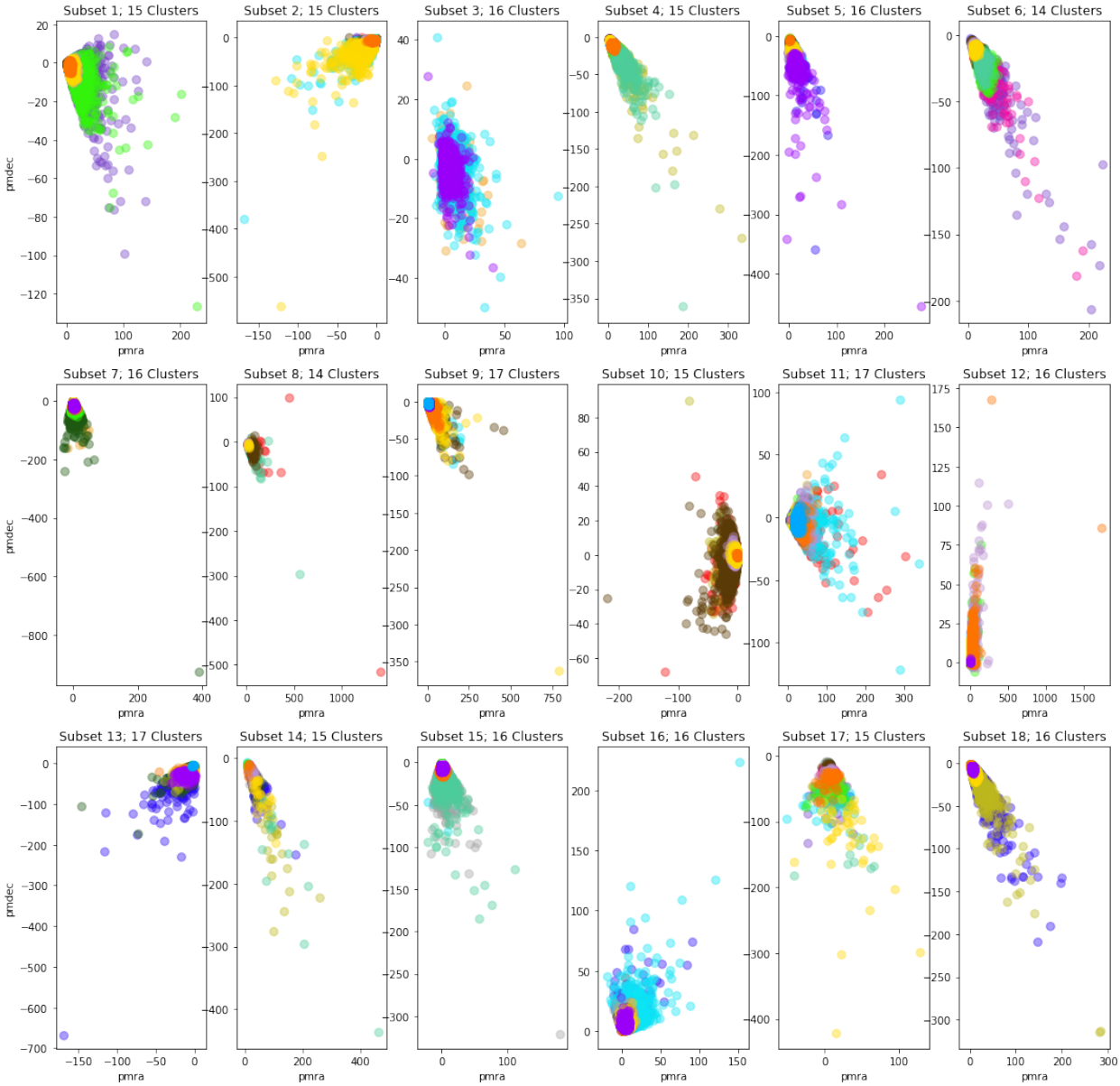

Fig. 18.— The clustering results for the two-step KMeans algorithm, shown in velocity space. Each subset is essentially one cluster from Figure 14, so even though the clusters in each subset appears more dispersed, it is confined to a small subsection of the velocity space, reducing overall dispersion.

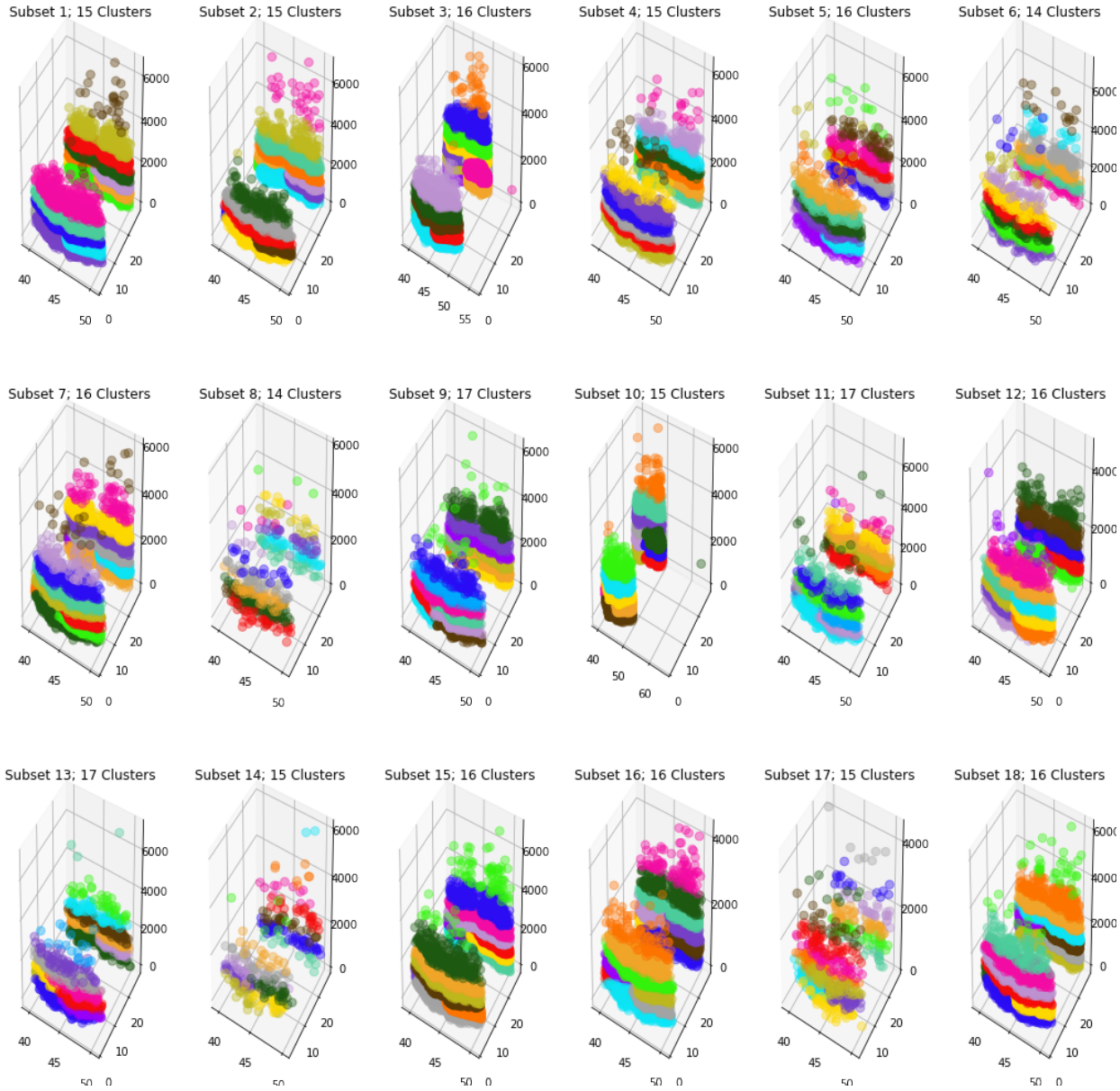

Fig. 19.— The clustering results for the two-step KMeans algorithm, shown in the three spatial dimensions. Most of the clusters are well contained in spatial dispersion, though the bleeding for the farthest clusters does appear in a few of the clusters, shown as two clusters of the same color.

a cluster, and there are outliers. Realistically, this is ideal for finding star clusters in Gaia data, because we don't know how many clusters to expect, and because we expect some number of sources to be field stars, which are not connected to any cluster.

There is a catch to this advantage, and that is how to define what makes up a cluster. DBSCAN requires you specify two parameters, which allow it to accurately define clusters in your data based on overdensities in your data. You have to give it a distance in which it looks for nearby neighbors for the clustering, as well as the minimum number of data points needed within that distance in order to create a cluster. The key is figuring out the proper balance of these parameters to get DBSCAN to create physically realistic clusters in the data.

The problem with Gaia data is that I am using 5 dimensions, most with differing units. So it becomes hard to define a distance that weighs these dimensions equally. To do so, I standardized all of the units of my data using sklearn's StandardScaler preprocessing function, transforming my data into standard units with which I can define a distance. However, by doing this, I lost the physical nature of my data; I would like to define distance based on physically realistic sizes of clusters, but that cannot be done in this fashion, so I had to choose my distance (and minimum number of points) and hope that the resulting clusters are realistic.

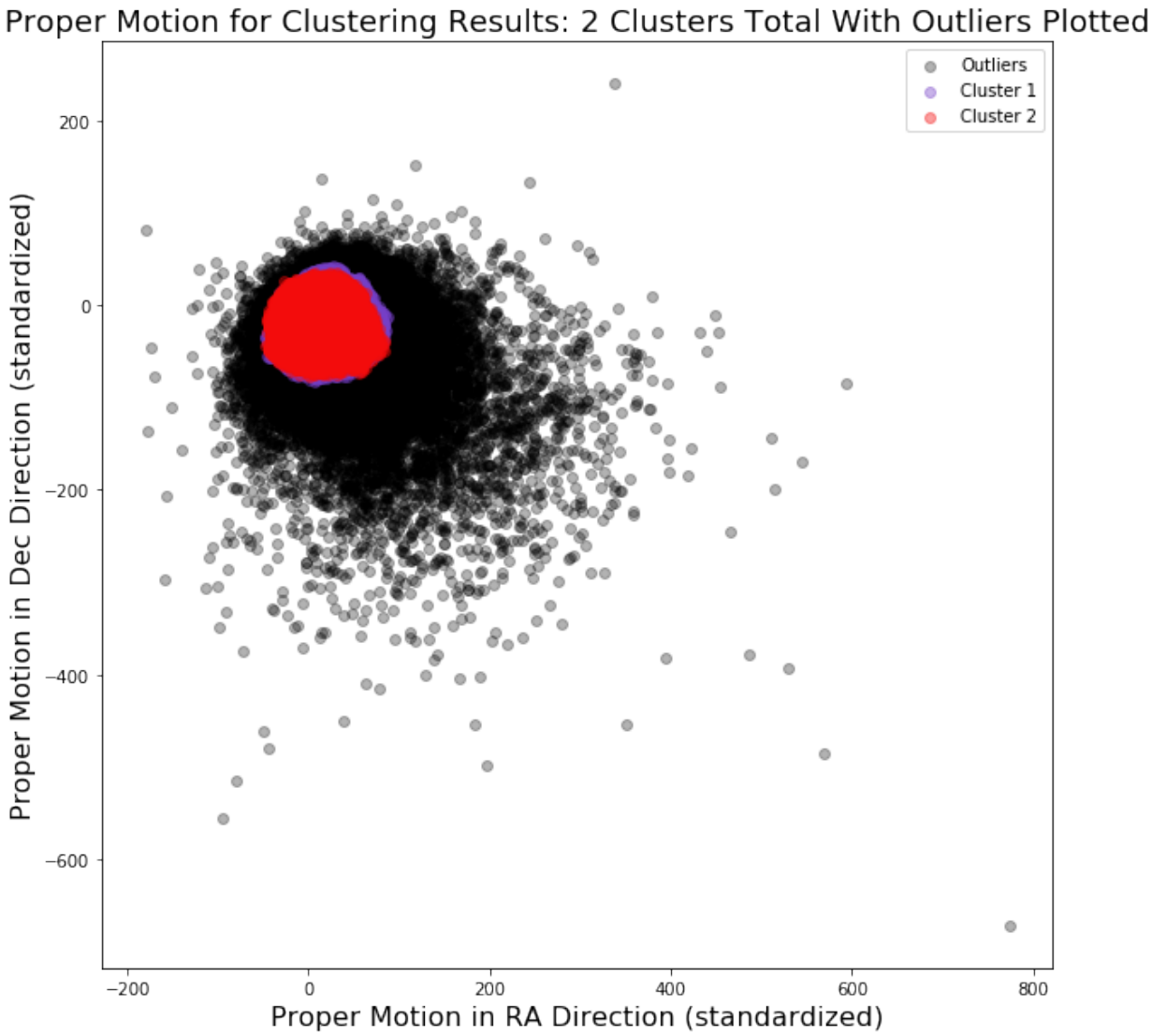

Fig. 20.— The clustering results for my second attempt with DBSCAN, which contained two large clusters (in red and purple) and an overwhelming plurality of outliers, shown in black. It can be seen that the clusters are with the lower velocity data points, while the higher velocity points are all outliers.

My first two attempts created two very large clusters containing a good number of points from each subset in my data, and either a few smaller clusters, or no small clusters at all. In both cases, a plurality of data points were found to be outliers, which in itself is not necessarily a problem. I was more interested in finding realistic clusters, and did not worry about the number of outliers found in the algorithm. Figure 20 illustrates the plurality of outliers in the data, and the compactness of the clusters in the velocity space.

On my third attempt at clustering with DBSCAN, I obtained better results, in which I only had three particularly larger clusters (which were not nearly as large as previous results), as well as a number of smaller, realistic looking clusters. The parameters for this run were an eps of 0.18 and a minimum sample number of 20. The resulting clusters are shown in Figures 21 and 22. There were a total of 77 clusters found, still with an overwhelming plurality of outliers, not plotted.

5. Discussion

5.1. Realistic or Unrealistic Clusters

Now that I utilized these clustering techniques, it's important to tie the results back to the original aim of the project by determining whether or

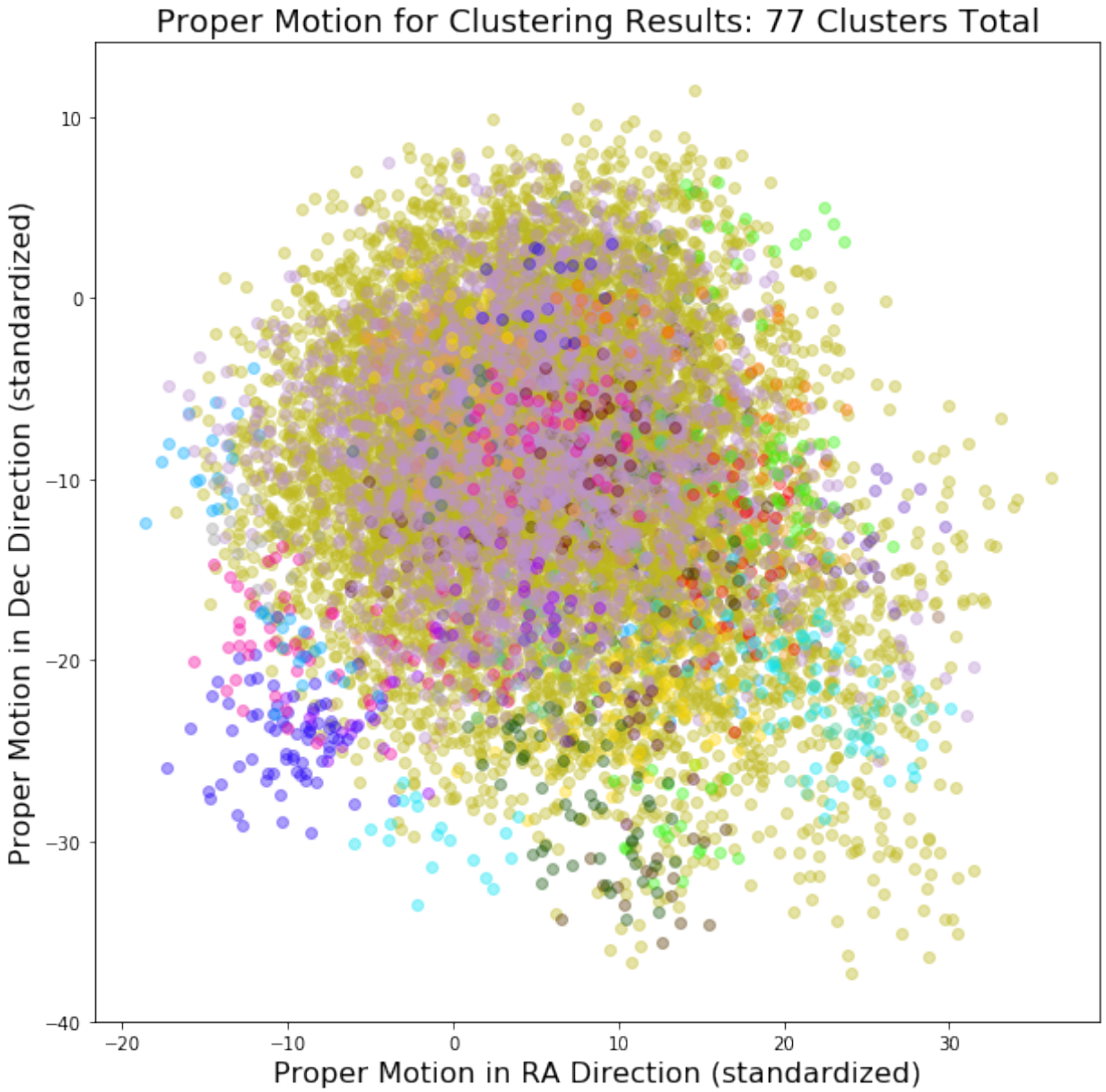

Fig. 21.— The clustering results for my final attempt with DBSCAN, which contained a few semi-large clusters and a number of smaller clusters, shown in velocity space. Despite the larger clusters, the smaller ones seem pretty well contained. Outliers are not plotted for clarity.

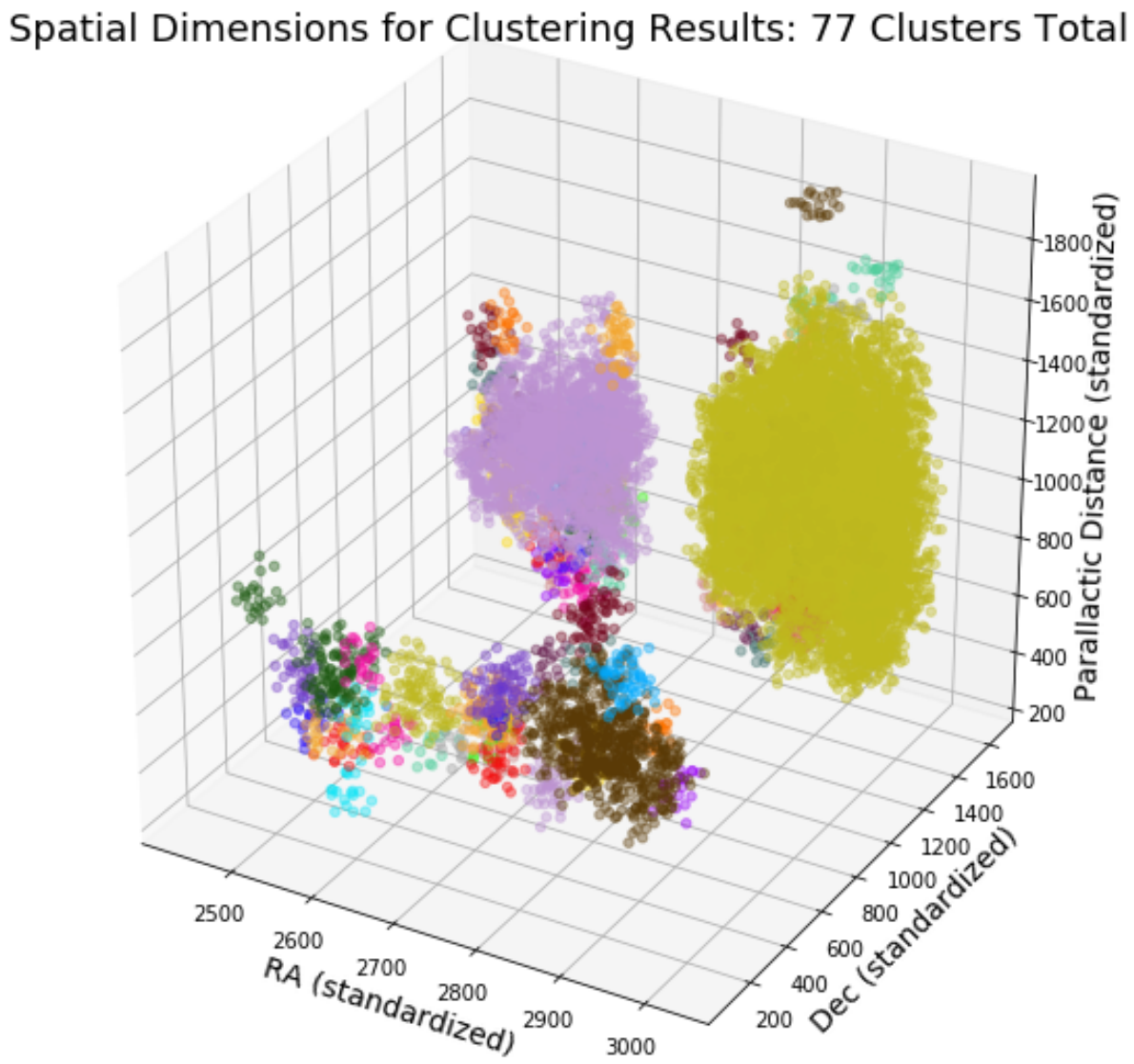

Fig. 22.— The clustering results for my final attempt with DBSCAN, which contained a few semi-large clusters and a number of smaller clusters, shown in the three spatial dimensions. Despite the larger clusters, the smaller ones seem pretty well contained. Outliers are not plotted for clarity.

not the techniques produced astronomically realistic stellar clusters. Given that the data is unlabeled, there aren't known clusters within my sample on which to base success, but I can use average cluster properties to get a baseline for success. Then, I can use these baselines to determine courses of action that may improve upon the methodology I employed.

5.1.1. Cluster Core Size

One way to determine success is to look at the size of the clusters. Cantat-Gaudin et al. (2018) did similar work of finding clusters in Gaia data, and employed a method of compactness for determining if the cluster size was realistic, stating that a cluster's "member stars must share common properties, and be more tightly distributed on the sky than a random distribution." They never get more specific about how this is defined, so I cannot reproduce their methods (and they use a proprietary code called UPMASK to do this analysis). However, this isn't the only way to get size comparisons. Open stellar clusters of the kind I'm hoping to discover generally have a central core, typically a few parsecs in radius, and an extended coma of lower density, which can extend out to ten or twenty parsecs. As such, I can use a measure Cantat-Gaudin et al. (2018) uses, median distance or r_{50} (radius below which 50% of the stars are located within) as a proxy for the core size.

For both KMeans and DBSCAN I calculated the median distance by determining each cluster member's distance from the cluster center. This task was very straightforward for KMeans, as implicit to the algorithm a cluster center is generated in each dimension. Given that I am working in a spherical coordinate system, I calculated the distance from the center using the formula in spherical coordinates:

$$\mathrm{d} = \sqrt{\mathrm{r}^2 + \mathrm{r'}^2 - 2\mathrm{rr'}(\sin(\theta)\sin(\theta')\cos(\phi - \phi') + \cos(\theta)\cos(\theta'))} \qquad (2)$$

where r and r' are parallactic distances in parsecs, θ and θ' are declinations in radians, and ϕ and ϕ' are right ascensions in radians. By calculating d for each cluster member, I found the distance distributions for each cluster in parsecs.

Calculating Equation 2 was more difficult for DBSCAN because cluster centers were not defined as part of the algorithm. As such, I created a cluster center by calculating the average RA, dec, and parallactic distance for the members. I then calculated the distance distribution as the distances of all cluster members from that average point.

Figure 23 shows the r_{50} distribution for my KMeans clusters. It is clear that these clusters are too large, as their cores should be no more than ten parsecs in size, yet the vast majority are much larger than that. A likely

reason for this is that KMeans doesn't have a built-in way of rejecting data points into outliers. By requiring all points to be cluster members, they will be artificially large. It is not obvious how to deal with this issue, as all of the points, including potential outliers are used to calculate cluster centers, and so removing them either before or after is tainting your results for cluster centers, and there isn't a good way to define outliers prior to running the algorithm itself, as we don't know where clusters might be located.

Figure 24 shows the r_{50} distribution for the DBSCAN clusters. In this case, while the sizes are still too large to be realistic, they are much more realistic than those from KMeans. In many cases, clusters were only too large by a factor of three, which is not bad. I think a good reason for this is that unlike KMeans, DBSCAN is potentially able to identify field stars (outliers), and so clusters can avoid becoming artificially inflated by far away data points. However, there is a caveat to the results from DBSCAN, which is that my two-subset sample created artificial boundaries in places where there normally wouldn't be one. The empty space between the two subsets is not normally empty, so it's hard to know how well the algorithm would keep clusters small if this boundary was not present. To mitigate this in the future, I could use larger computing resources allowing for the full dataset to be utilized for modeling and analysis.

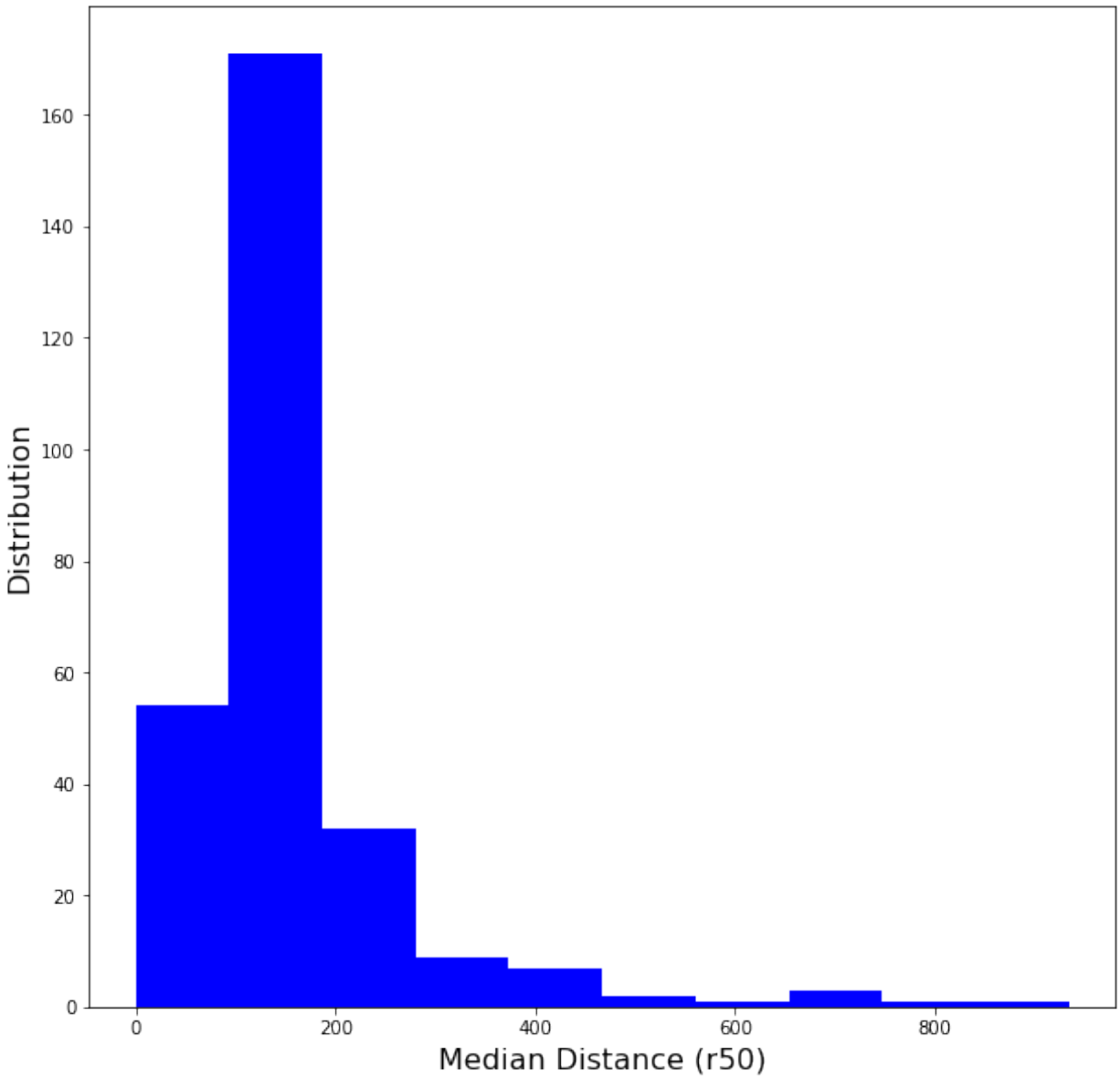

Fig. 23.— The median distance (r_{50}) distribution for my KMeans clusters. These distances are too large to be realistic, as typical core sizes are fewer than ten parsecs across.

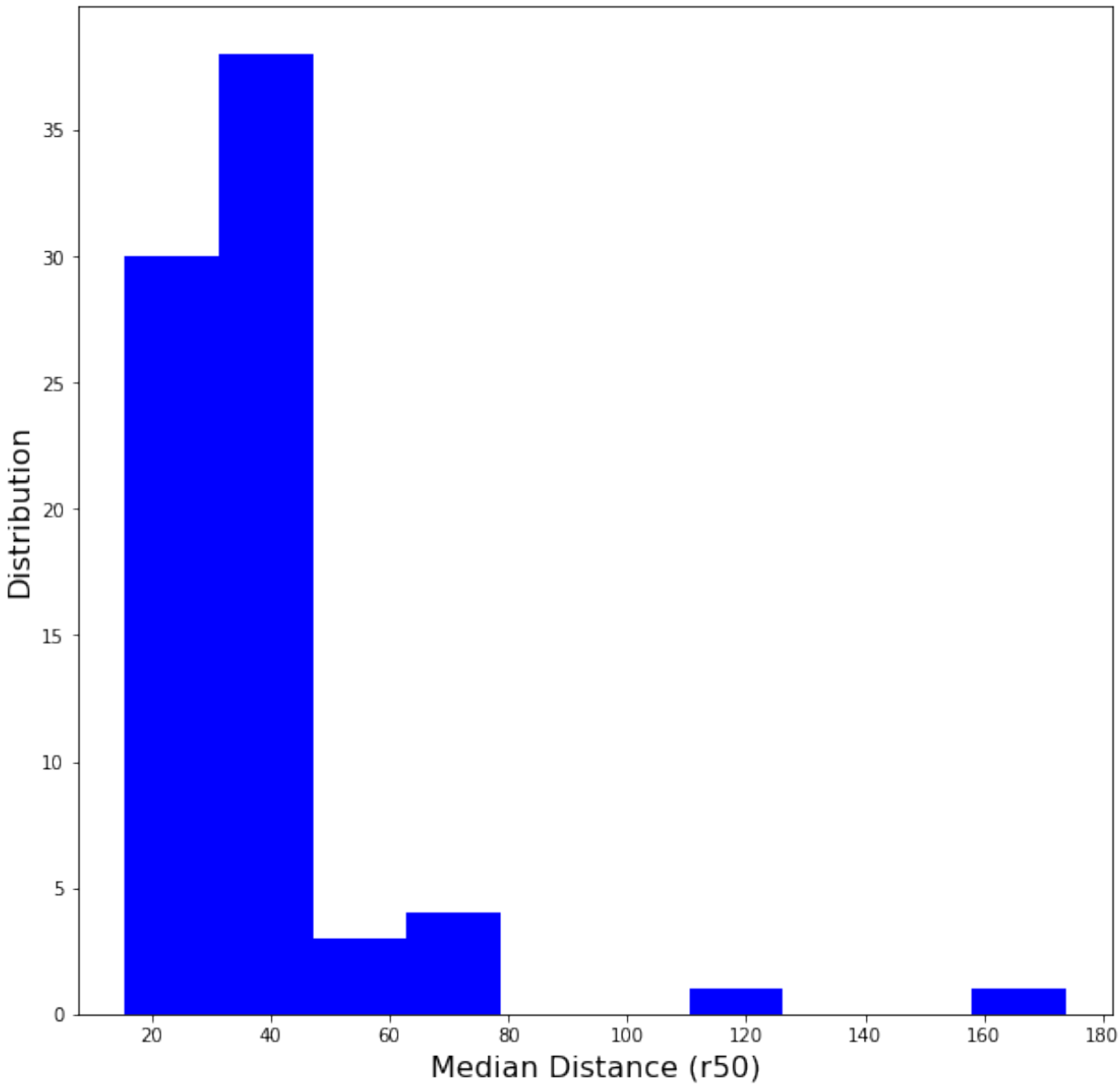

Fig. 24.— The median distance (r_{50}) distribution for my DBSCAN clusters. These distances are also too large to be realistic, as typical core sizes are fewer than ten parsecs across, but these are much more realistic than those identified using KMeans.

5.1.2. Velocity Dispersion

Another measurement I can use for comparison is the velocity dispersions of the clusters. Kuhn et al. (2019) gives a good estimate for an average velocity dispersion of a few kilometers per second for stellar clusters. Dispersions are easy to calculate, as I can just take the standard deviation of the velocities for each cluster, making sure they are in the correct units of kilometers per second.

The velocity dispersions in the RA and Dec directions for my KMeans clusters are plotted in Figure 25. The plot illustrates that the velocity dispersions are too high for my KMeans clusters in order to be physically realistic. Like the spatial dispersions, I suspect this can be improved by somehow identifying outliers in the clusters.

Figure 26 shows the velocity dispersions for my DBSCAN clusters. These dispersions, though in many cases still too high, are much smaller than for KMeans, and in many cases have realistic values much similar to those you might expect. Given these values, and the spatial dispersions, I think it can be said that the DBSCAN algorithm was much better than KMeans at determining more realistic stellar clusters, though more work still would need to be done in the future.

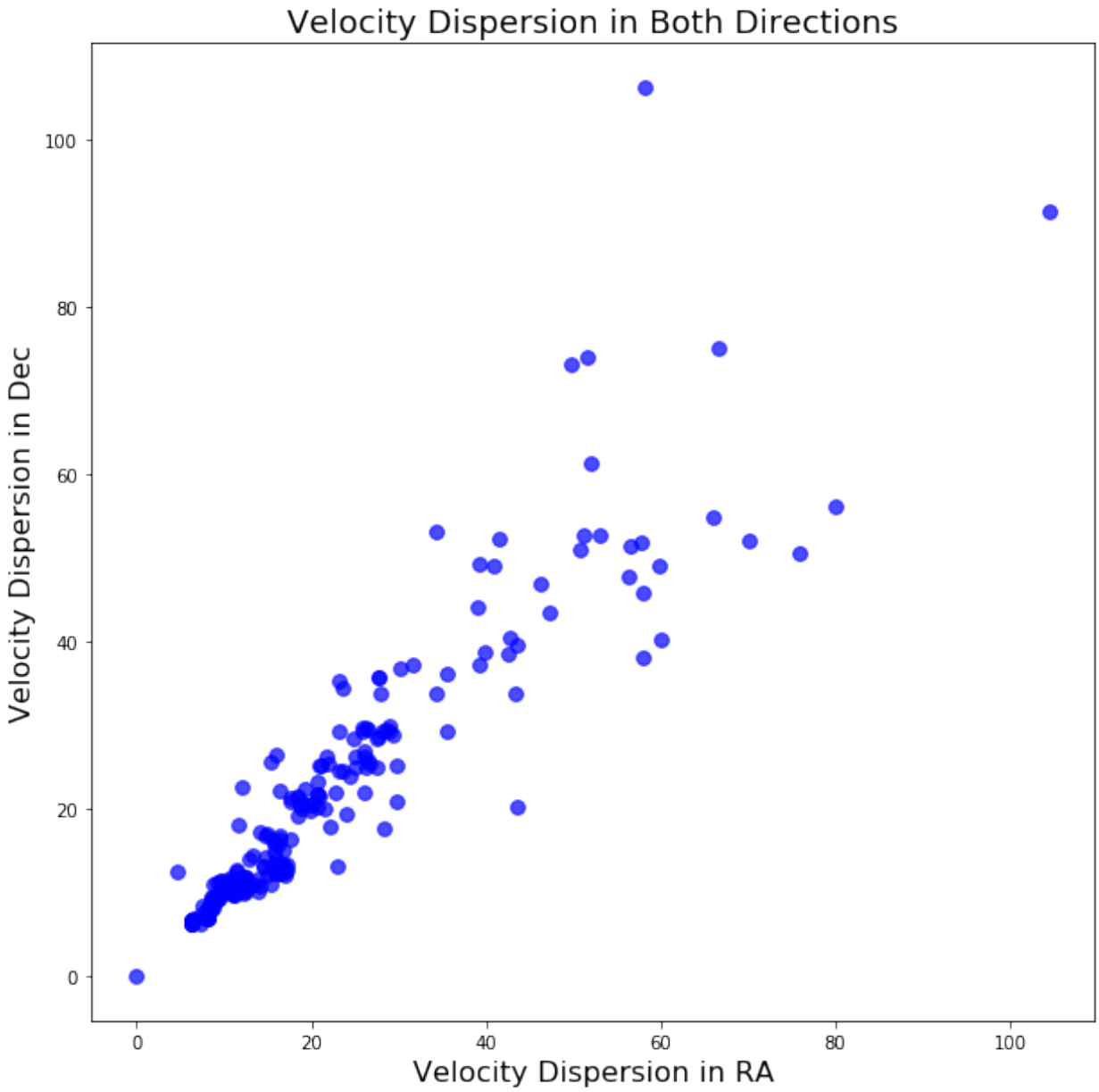

Fig. 25.— The velocity dispersions for my KMeans clusters, in each tangential direction. Dispersions are given here in kilometers per second. Typical velocity dispersions are only a few kilometers per second, making these dispersions too large to be physical. This is likely another result of the fact that KMeans does not identify outliers. For KMeans, there is a clear, positive correlation between the dispersions in each direction.

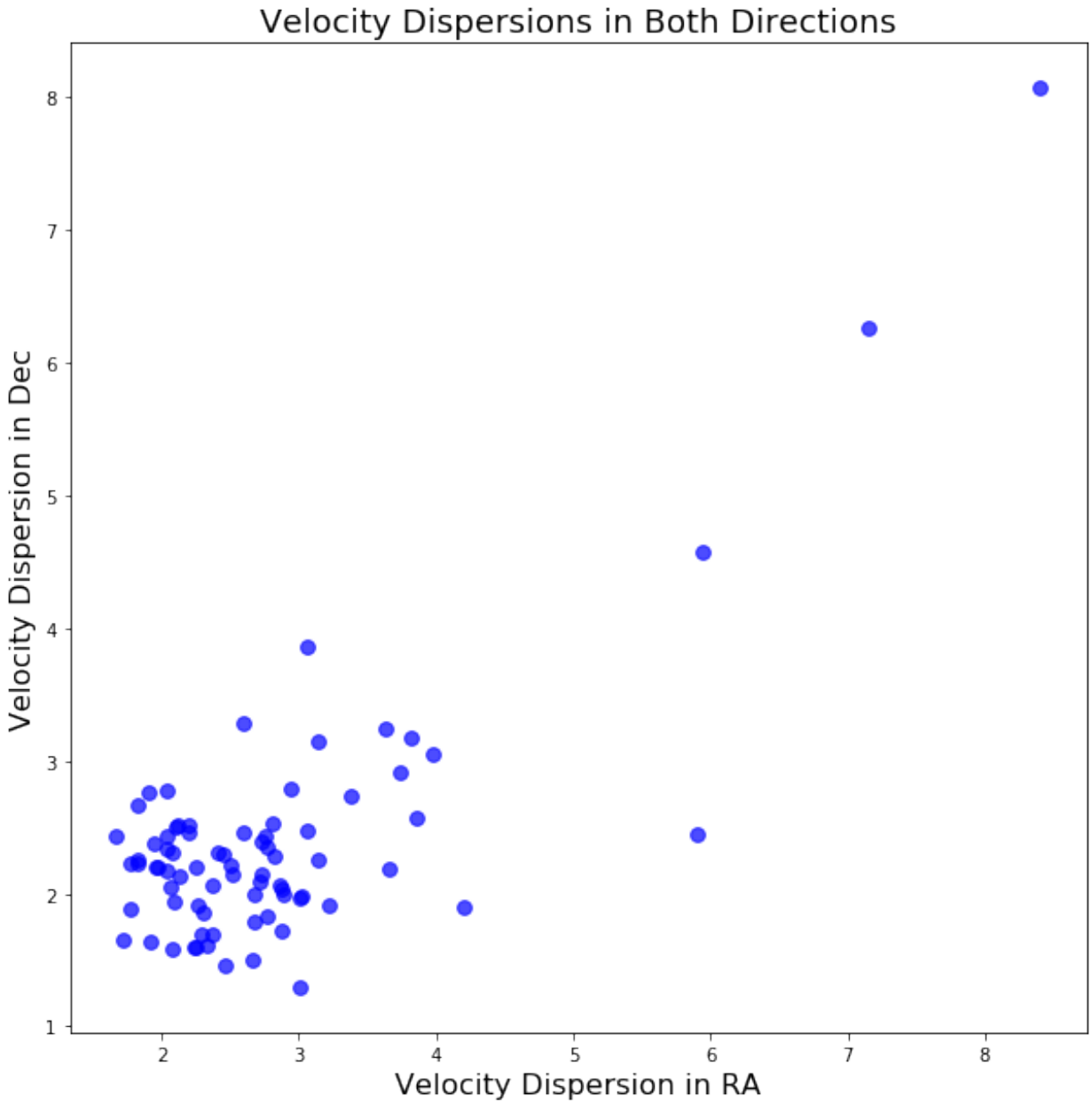

Fig. 26.— The velocity dispersions for my DBSCAN clusters, in each tangential direction. Dispersions are given here in kilometers per second. Typical velocity dispersions are only a few kilometers per second. While many of these clusters have larger dispersions than desired, a fair number of them are of the correct order of magnitude to be realistic. Similar to KMeans, there is a positive correlation between the dispersions in each direction.

5.2. Distribution Correlations

One additional thing I looked into after completing my modeling was to see whether or not the clusters identified by the two algorithms showed any correlations with each other. The first correlation I looked into was whether or not the core size (r_{50}) was correlated with the number of members in the cluster (cluster size). Figure 27 shows a scatter plot of those two variables for the KMeans clusters. Interestingly, there appears to be an inverse-proportionality between the core size and cluster size for the KMeans clusters. I would normally expect the opposite to be true–that as a cluster gained more members, it would spread out. However, I think this can be explained by my data and the nature of KMeans. In this case, the densest areas in the data would end up as small clusters with large members, while the data in less dense outskirts, rather than being outliers, become large clusters that ultimately don't have many members. So this creates the inverse-proportional shape.

Figure 28 shows the relationship between core size and cluster size for my DBSCAN clusters. Here, the opposite is true compared to that of KMeans; we see a clearly positive correlation for the smaller clusters, with some deviation from the few large clusters present. This is much more what I would normally expect for cluster behavior. I think DBSCAN's method is

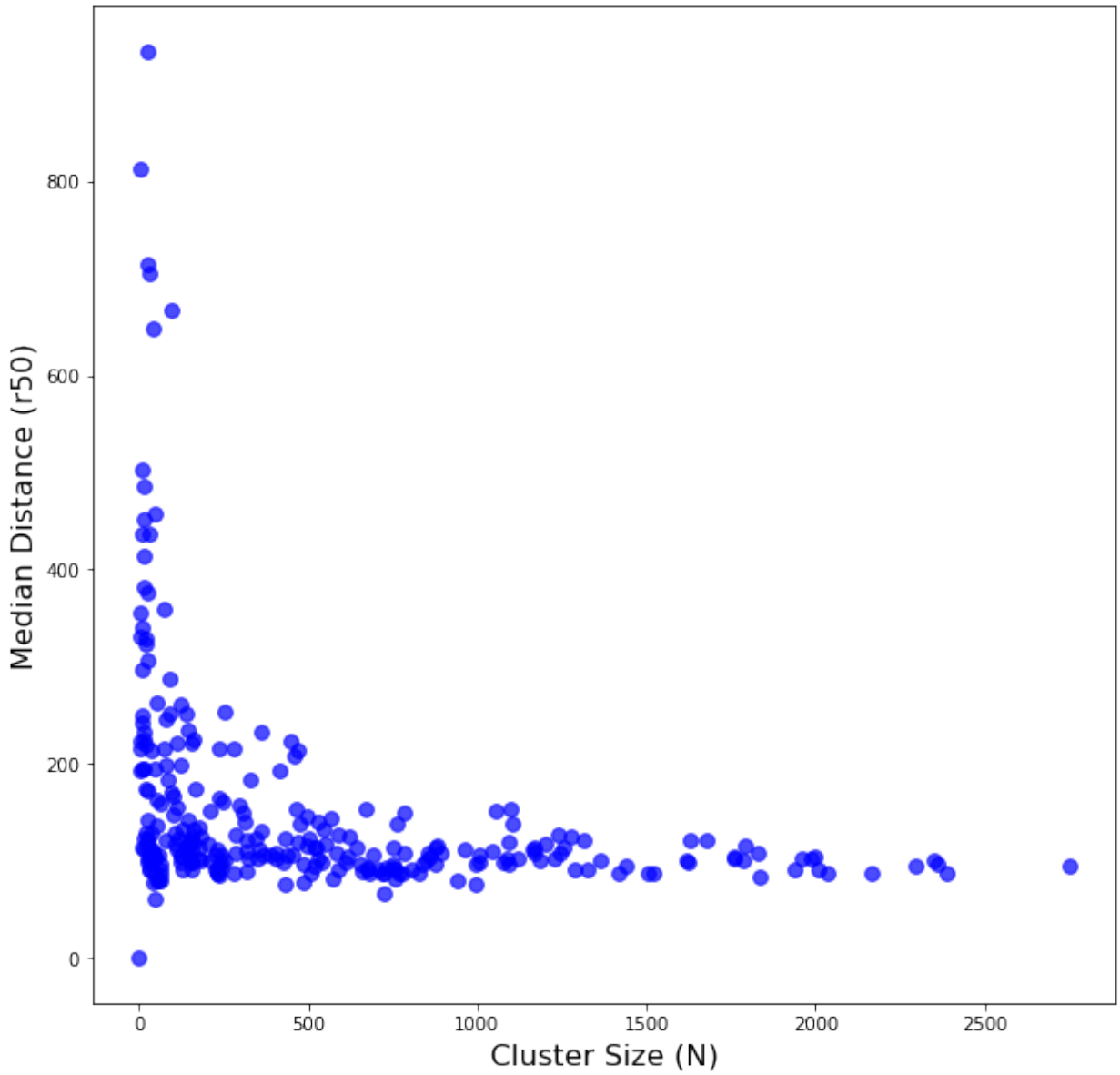

Fig. 27.— The velocity dispersions for my DBSCAN clusters, in each tangential direction. Dispersions are given here in kilometers per second. Typical velocity dispersions are only a few kilometers per second. While many of these clusters have larger dispersions than desired, a fair number of them are of the correct order of magnitude to be realistic. Similar to KMeans, there is a positive correlation between the dispersions in each direction.

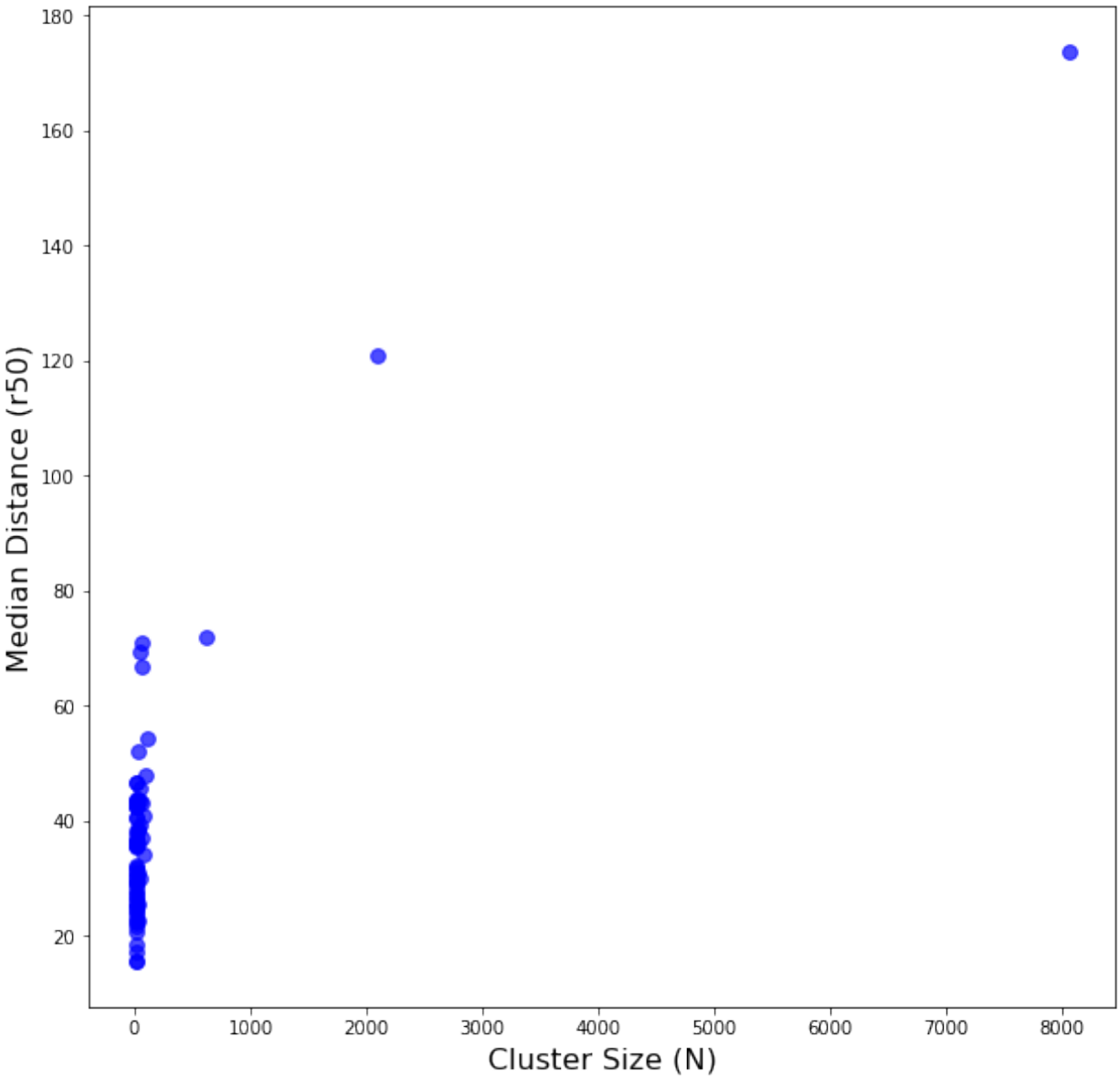

Fig. 28.— The velocity dispersions for my DBSCAN clusters, in each tangential direction. Dispersions are given here in kilometers per second. Typical velocity dispersions are only a few kilometers per second. While many of these clusters have larger dispersions than desired, a fair number of them are of the correct order of magnitude to be realistic. Similar to KMeans, there is a positive correlation between the dispersions in each direction.

more accurate because it is producing more accurate clusters themselves.

I then looked into the relationship between the velocity dispersions and the cluster sizes. For both KMeans and DBSCAN, the correlations are very similar to that of the core sizes versus cluster sizes, shown in Figures 29 and 30, respectively. The reasons behind these correlations and anti-correlations are likely for the same reasons as the previous ones. In order to quantify these correlations, I calculated the pearson correlation coefficients for each of these relationships, given in Table 2. In all cases, the p-values were so low that they were essentially zero.

6. Conclusions, Future Work, and Recommendations for the Client

For this project, I utilized over 140,000 sources from the Gaia DR2 in order to determine the effectiveness of using KMeans and DBSCAN clustering algorithms in discovering stellar clusters. I downloaded the relevant data from the Gaia query service and archive, obtaining data for the three spatial dimensions and two tangential velocity dimensions.

Given the dataset I used, I found that the total tangential velocity distributions were log-normal, and that the two subsets of data I used were characterized by log-normal distributions of different parameters. I found two hypervelocity stars in the dataset, as well as a number of high velocity

Features	(Pearson r, p-value)		
	All Stars	Main Sequence Only	Giants
Distance vs. Luminosity	(0.164, 0.000)	(0.263, 0.000)	(0.063, 0.062)
Effective T vs. Luminosity	(-0.063, 0.000)	(0.447, 0.000)	(-0.499, 0.000)
Distance vs. Effective T	(0.300, 0.000)	(0.356, 0.000)	(0.115, 0.001)

Table 1: The Pearson Correlation Coefficients (r) for the stellar features and distance, as well as their p-values, for all stars in the sample, main-sequence stars, and giants. All of these values were calculated using the scipy.stats function pearsonr. Most of the p-values were small enough to be effectively zero.

Features	(Pearson r, p-value)	
	KMeans	DBSCAN
Cluster Size vs. Core Size	(-0.330, 0.000)	(0.822, 0.000)
Cluster Size vs. Velocity Disp in RA	(-0.506, 0.000)	(0.692, 0.000)
Cluster Size vs. Velocity Disp in Dec	(-0.509, 0.000)	(0.802, 0.000)
Velocity Disp in Dec vs. Velocity Disp in RA	(0.923, 0.000)	(0.771, 0.000)

Table 2: The Pearson Correlation Coefficients (r) for the cluster features, as well as their p-values, for all clusters identified using the KMeans and DBSCAN algorithms. All of these values were calculated using the scipy.stats function pearsonr. All of the p-values were small enough to be effectively zero.

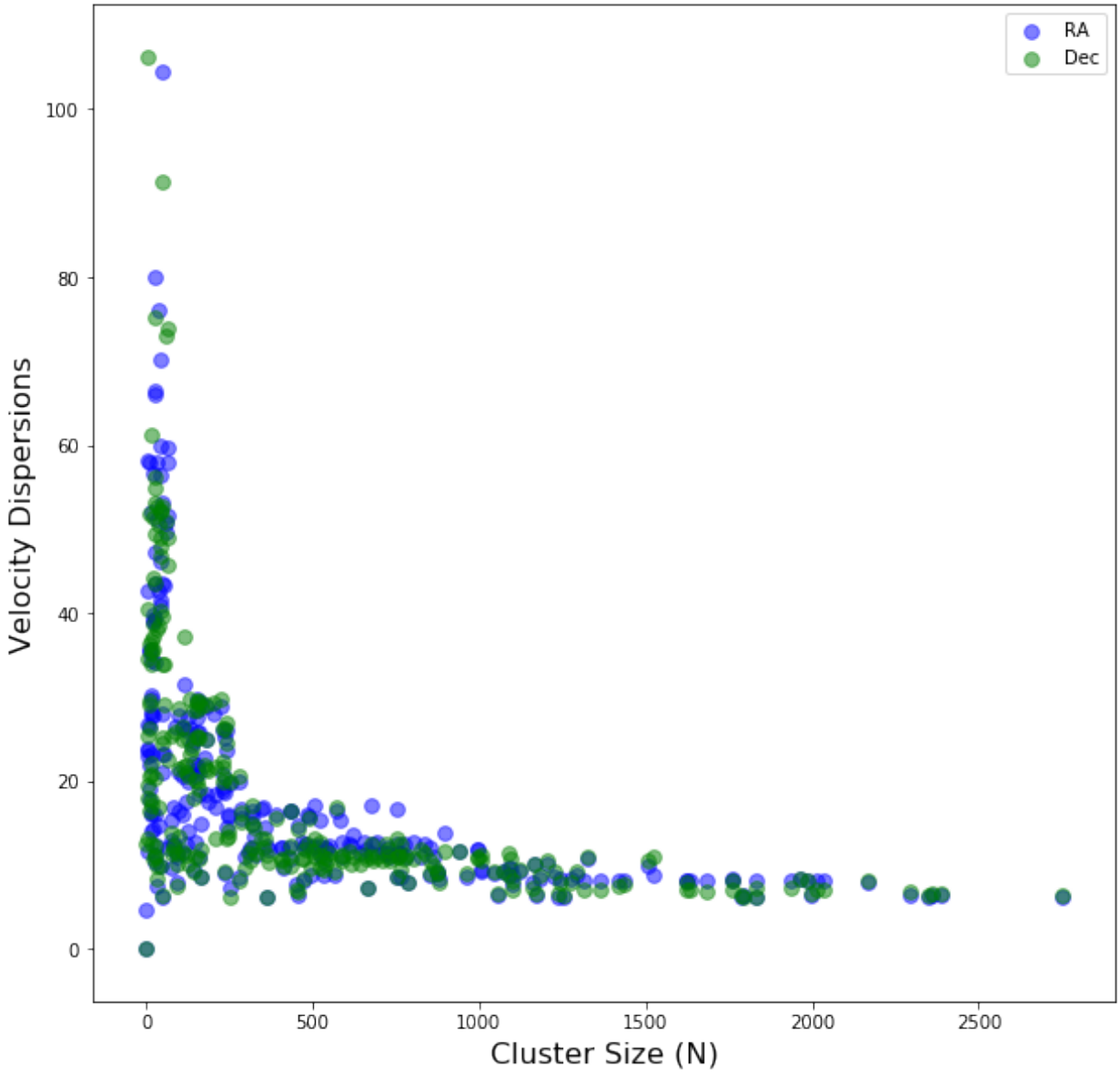

Fig. 29.— The cluster sizes for my KMeans clusters plotted against the velocity dispersions. Dispersions are given here in kilometers per second. This relationship closely follows that between the core size and cluster size, likely for similar reasons.

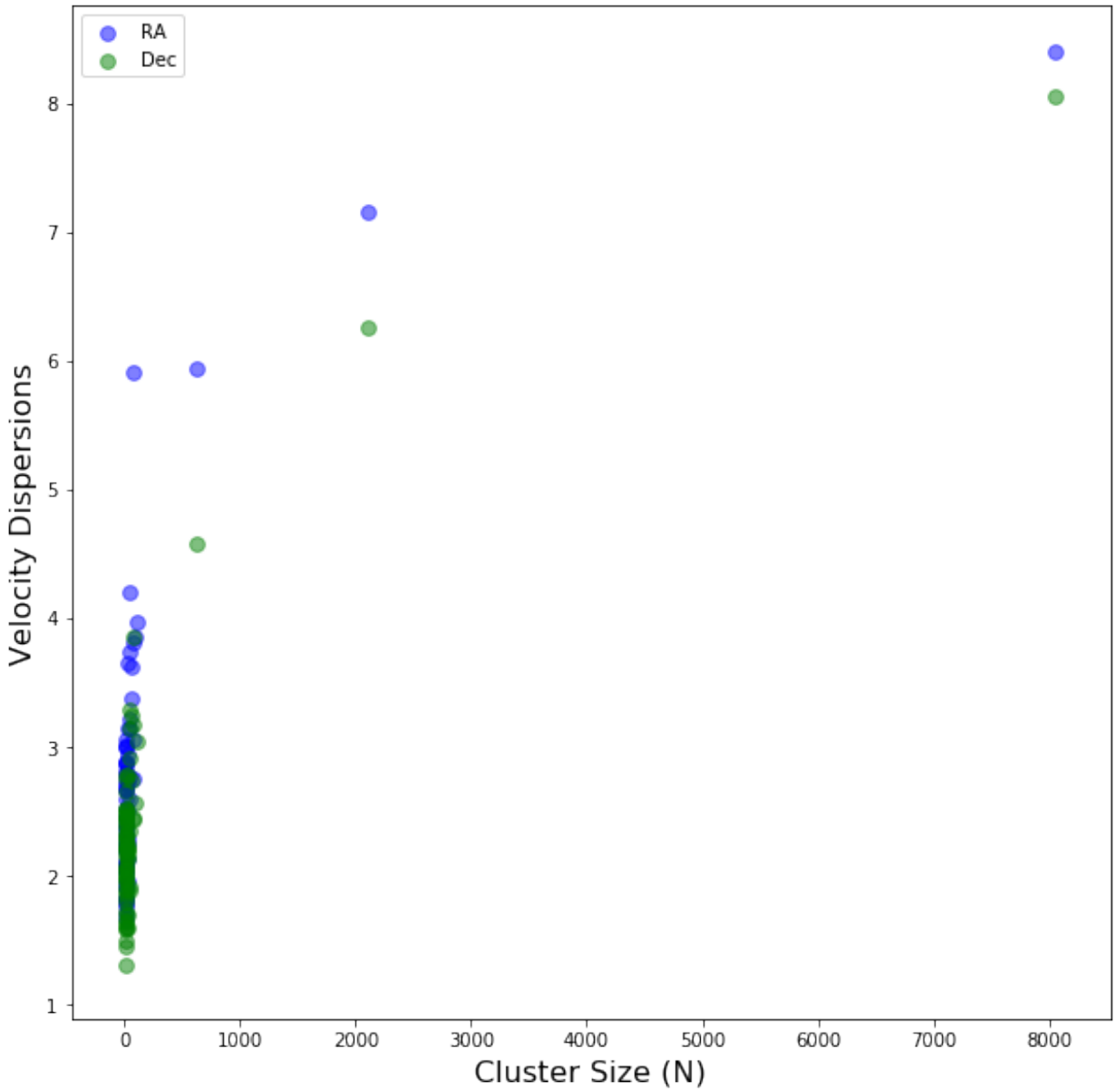

Fig. 30.— The cluster sizes for my DBSCAN clusters plotted against the velocity dispersions. Dispersions are given here in kilometers per second. This relationship closely follows that between the core size and cluster size, likely for similar reasons.

stars. Of the sources containing stellar parameter data, I was able to separate out the giants from the main sequence stars, and verify the selection bias that smaller main-sequence stars are not found at larger distances. I was also able to determine that the giant stars have a different tangential velocity distribution than main-sequence stars in the sample.

For the KMeans clustering algorithm, I found the best results by performing the clustering in two separate steps: first running a KMeans clustering on the 2-D velocity data only, and then taking each of those clusters and running a second KMeans clustering using the 3-D spatial data. This allowed for more realistic clusters to be found using the algorithm, totaling 281 clusters. However, the results showed that even when this was performed, the clusters were too large and had dispersions that were too high. In the future, a way of rejecting outliers (i.e., determining field stars) should help to mitigate or possibly eliminate this problem.

As for DBSCAN, the algorithm only required one step in the process, but the data had to be standardized, which caused a loss in physical information. Nevertheless, the resulting 77 clusters were mostly more realistic than those from KMeans (with a few exceptions), likely due to the algorithm's ability to reject outliers. The trends shown between these clusters and their parameters are also likely more realistic as a result. However, by breaking my dataset into two subsets of Gaia sources, I introduced false boundaries,

which could have created issues for DBSCAN.

For future work, it would be a generally good idea to utilize more of the dataset for this kind of analysis. Especially for DBSCAN, using a continuous dataset (or even the entire 1.3 billion sources) would allow for modeling without falsely generated boundaries. This would allow for a better examination of whether or not this technique can still generate realistic clusters. In order to carry out this work with a larger dataset, more computing power than I have on my laptop is necessary. Utilizing Spark and cloud computing resources, such an endeavor should be possible, especially for DBSCAN (as it is more scalable than KMeans is).

Utilizing the full Gaia DR2 would also allow for greater exploration of the patterns I saw in the data regarding stellar and cluster properties, and would lead to identification of more hypervelocity stars. With such a larger sample, more insights could be discovered through statistical analyses of these subsamples compared to the overall population of stars.

Given that the client for this project is any researcher hoping to find clusters in this or any other similar dataset using machine learning techniques, I think the best recommendation is to consider these clustering techniques carefully. There is certainly a lot of promise in these algorithms, and with the future work outlined above, it will be possible to really

determine the feasibility and advantages of these techniques compared with any current ones. Having access to a supercomputing cluster would also be a big help given the scale of this type of work, and many academic and research institutions have access to these, so I would recommend this as well.

Gaia's DR2 (and future data releases) clearly has the potential to change our understanding of nearby and intermediate distance stellar clusters, and unsupervised clustering techniques seem like they could be very useful in uncovering these new insights. It's an exciting time to be a stellar astronomer!

Acknowledgements

This work has made use of data from the European Space Agency (ESA) mission *Gaia*[6] (`https://www.cosmos.esa.int/gaia`), processed by the *Gaia* Data Processing and Analysis Consortium (DPAC, `https://www.cosmos.esa.int/web/gaia/dpac/consortium`). Funding for the DPAC has been provided by national institutions, in particular the institutions participating in the *Gaia* Multilateral Agreement.

[6]https://gea.esac.esa.int/archive/documentation/GDR2/Miscellaneous/sec_credit_and_citation_instructions/

REFERENCES

Bailer-Jones, C. A. L., Rybizki, J., Fouesneau, M., Mantelet, G., & Andrae, R. 2018, AJ, 156, 58

Cantat-Gaudin, T., Jordi, C., Vallenari, A., et al. 2018, A&A, 618, A93

Gaia Collaboration, Prusti, T., de Bruijne, J. H. J., et al. 2016, A&A, 595, A1

Gaia Collaboration, Brown, A. G. A., Vallenari, A., et al. 2018, A&A, 616, A1

Kuhn, M. A., Hillenbrand, L. A., Sills, A., Feigelson, E. D., & Getman, K. V. 2019, ApJ, 870, 32