Draft version May 3, 2023
Typeset using LaTeX default style in AASTeX631

Small Exoplanet Classification with Clustering Machine Learning Algorithms

Lorraine Nicholson[1]

[1] *University of Florida, Department of Astronomy*

1. INTRODUCTION

Since the launch of Kepler in 2009, thousands of transiting exoplanets have been discovered. The transit method if powerful because not only does it reveal that a planet exists, it also gives us some information about the planets radius and orbital period. These discoveries have revealed that small planets (<4 R⊕) are quite common in the galaxy (Borucki et al. 2010). More surprisingly, there is an apparent lack of planets between 1.5 - 2.0 R⊕ (Owen & Wu 2017; Fulton et al. 2017; Van Eylen et al. 2018). Furthermore, the planets smaller than 1.5 R⊕ appear to have very high densities, consistent with rocky Earth-like planets. These planets are characterized as "super-Earths" (Rogers 2015; Dressing et al. 2015). While the planets larger than 2.0 R⊕ tend to have low densities which are consistent with gaseous planets that have large H/He dominated atmospheres. These larger planets are characterized as "sub-Neptunes" (Lopez & Fortney 2014; Jontof-Hutter et al. 2016). The gap separating these two distinct groups is deemed the "radius valley." It is unclear what physical mechanism leads to the radius valley; hypotheses include core-powered mass-loss (Gupta & Schlichting 2019, 2022) and photoevaporation (Owen & Wu 2017).

The radius valley provides a unique method to be able to classify an exoplanet as super-Earth or sub-Neptune based on just its orbital period and radius. As opposed to the more traditional method based on it's density. I think this is important because to know a planets density we would also need some information about its mass, which we can't get from a transit measurement. The planets mass comes from a separate type of measurement, usually radial velocity. Now, we can classify these small planets with just the information received from the transit which makes it possible to classify even more planets than before.

Recent studies have aimed to use machine learning to determine the slope and overall shape of the radius valley. Specifically, Van Eylen et al. (2018) employed Support Vector Machine (SVM) learning to cleverly determine the slope of the radius valley. This was a smart choice since the data is linearly separable in log(planet radius)-log(orbital period) space, and SVM works great with linearly separable data. Additionally, SVM is able to output the equation of the line which separates groups of data. This study concluded that the equation of the radius valley is:

$$\log_{10}(R_P) = -0.10 \log_{10}(P) + 0.38 \tag{1}$$

where R_P is the radius of the planet and P is the orbital period of the planet. Importantly, the slope of the radius valley (-0.10) is negative and not very steep. Planets with $\log_{10}(R_P) > -0.10 \log_{10}(P) + 0.38$ are sub-Neptunes, whereas $\log_{10}(R_P) < -0.10 \log_{10}(P) + 0.38$ are super-Earths.

While SVM does a great job of separating the data, the aim of this project is to use unsupervised clustering algorithms to classify the two groups of planets. The reasoning is that by eye the data appears to be clustered into two distinct groups. The goal was to see if clustering algorithms can correctly recover the two groups. To this end I explored the performance of 3 different clustering algorithms: K-means clustering, Gaussian Mixture Model, and DBSCAN. These models did not perform as well as I would have liked, with accuracy's generally < 95%. I also tried a supervised Random Forest model which was able to correctly classify data with 100% accuracy. However, Random Forest does not have the added benefit of spitting out the equation of the radius valley the way that SVM was able to. Based on these results, unsupervised clustering algorithms seems to over complicate this problem that a simple SVM could solve really well.

2. METHODS

I obtained the training data set from Van Eylen et al. (2018) which included planetary radius and orbital period for 115 Kepler objects. I trained 3 different clustering algorithms and one random forest model, which I will explain

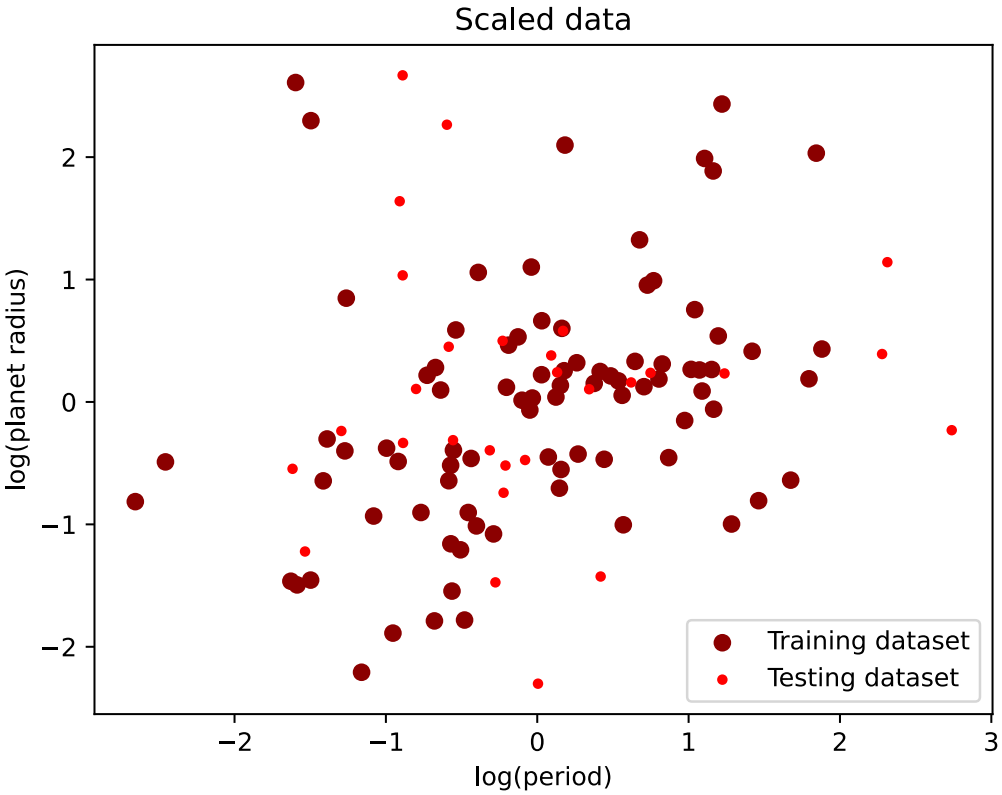

Figure 1. The entire data set, feature scaled, and split into training (big, dark red) and testing (small, red).

in more detail in the following subsections. For each model, I fed in two features; planet radius and orbital period. I expected to retrieve two classes (0 or 1) corresponding to either a super-Earth or a sub-Neptune.

I followed a similar set-up for each model for consistency. First, I prepared the data by performing feature scaling and I split the data randomly into 75% training and 25% testing. The final training and testing data set used for each model (except DBSCAN see section 2.3) is shown in Figure 1. Importantly, both the training and testing data set are a good representation of the data overall.

Although the clustering algorithms are unsupervised, I manually labeled the data as part of the preparation step to use for accuracy calculations later on. I labeled the data based on the equation of the radius-valley determined by Van Eylen et al. (2018) (see Equation 1). Specifically $\log_{10}(R_P) > -0.10\log_{10}(P) + 0.38$ is labeled as a sub-Neptune or "1". And $\log_{10}(R_P) < -0.10\log_{10}(P) + 0.38$ is labeled as a super-Earth or "0".

After preparing the data, I picked the necessary hyperparameters (see sections 2.1, 2.2, 2.3) and then trained and fit the model on the training data set only. I evaluated the performance by making predictions of the testing data set, and computing an accuracy score based on my manual labels. At this point, I investigated where things went wrong and tweaked hyperparameters when necessary. Finally, I compared my clusters to the classifications predicted by SVM in Van Eylen et al. (2018). In the following subsections I explain the nuances of each model.

2.1. *K-means Clustering*

The first algorithm I tried was sklearn's KMeans (Pedregosa et al. 2011). K-means clustering is an unsupervised machine learning algorithm used for grouping similar data points into K clusters. First, the user must choose the number of clusters, K, they want to find. I chose K=2 clusters because I want to recover two distinct groups; super-Earths and sub-Neptunes. From there, the algorithm selects initial centroid points from the data set. The distance of each data point to the centroids is calculated and each point is assigned a cluster based on which centroid it is

closest to. Then, a new mean centroid is calculated based on the clusters. The last two steps are repeated 10 times (n_init=10) before determining the final centroids and clusters. The final centroids are shown in Figure 2.

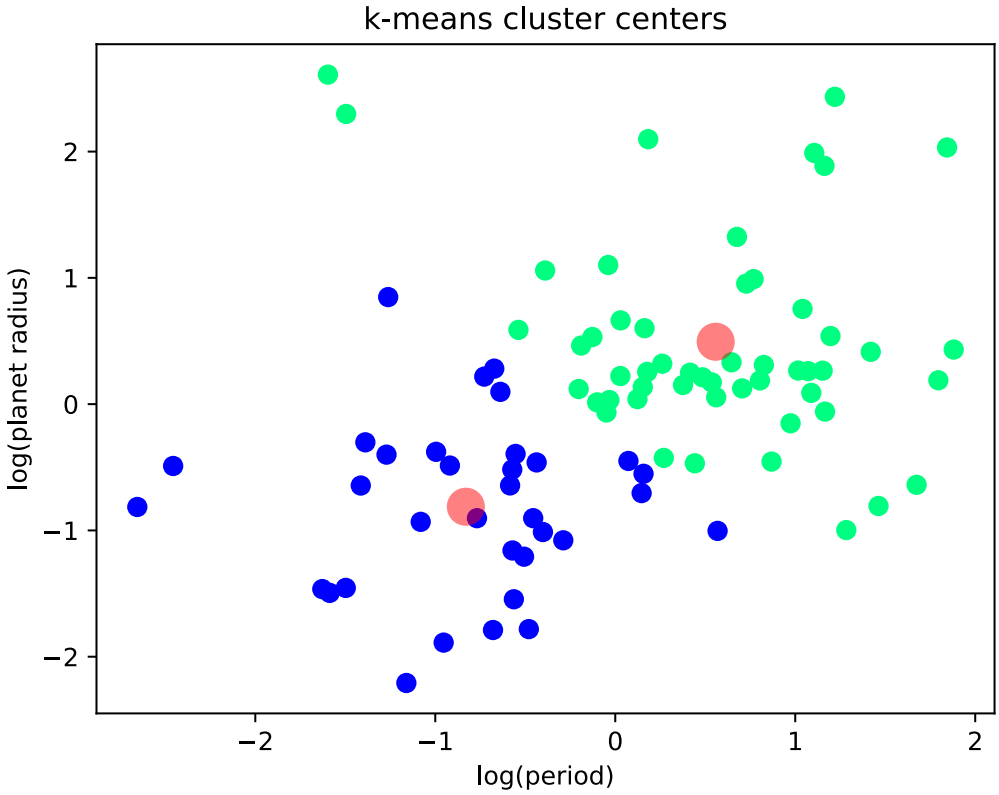

Figure 2. K-means final clusters and cluster centers (big red circles) for K=2 clusters after 10 iterations.

2.2. *Gaussian Mixture Model*

The next clustering algorithm I utilized was sklearn's Gaussian Mixture Model (GMM) (Pedregosa et al. 2011). It differs from K-means in that each cluster is represented by a probability density function, and therefore provides some information about the likelihood that a data point belongs to a cluster. Similarly to K-means, the user must input K number of clusters as a hyperparameter. I chose K=2 clusters for the same reasoning in section 2.1 and let it run for 10 iterations before assigning final clusters. The final decision boundary, cluster centers, and probability density function is shown in Figure 3.

2.3. *DBSCAN*

The final clustering algorithm I employed was sklearn's DBSCAN (Pedregosa et al. 2011). DBSCAN is a density-based algorithm which finds clusters within a data set. This algorithm determines clusters by drawing a circle of radius ϵ around each data point. If there exists a minimum number, min_samples, of data points within the circle then the point is considered to be a "core sample". All core samples are part of a cluster. Data points that neighbor core samples but do not meet the criteria themselves are considered to be boundaries of the cluster. Finally, data points that neighbor boundary points and do not meet the core sample criteria are assigned to be noise. DBSCAN is unique in that it can find noise in a data set, although this isn't very helpful for this project in which all the data is real.

I trained and tested this model on the entire data set, unlike the train-test split I did for all the other models. This is because DBSCAN does not have the same predicting ability as the other models, since clusters may change every time you introduce a new data point. For this project, I chose $\epsilon = 0.4$ and min_samples = 10. This decision was based

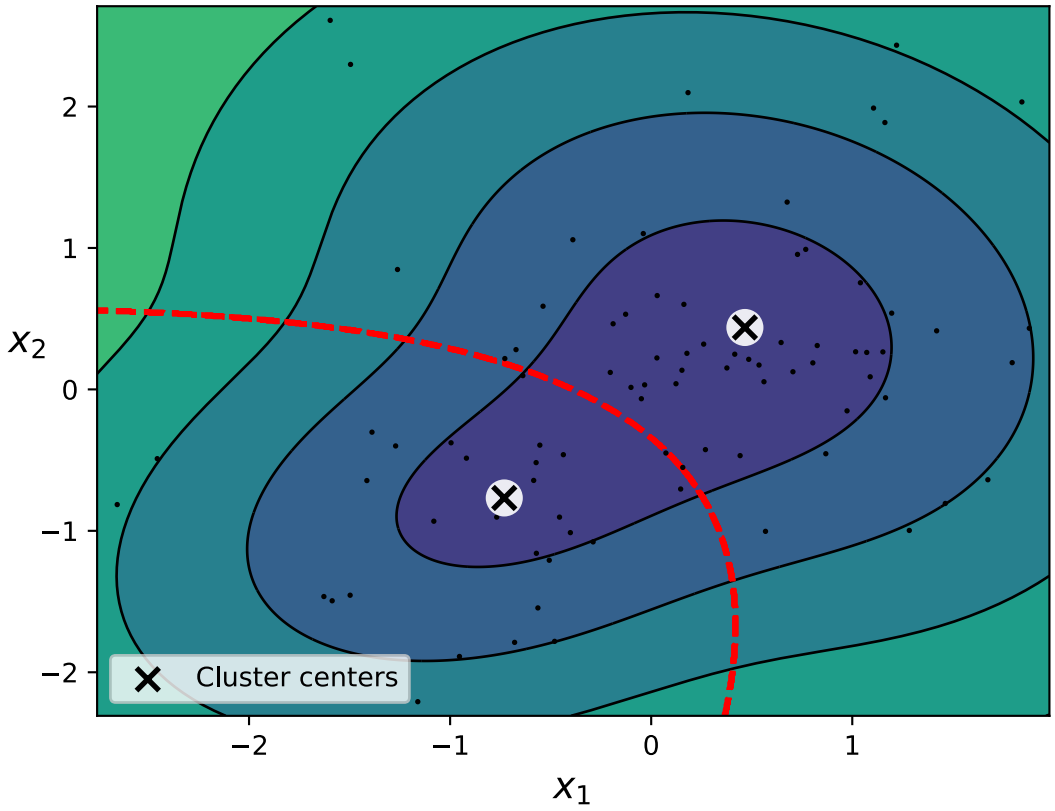

Figure 3. Gaussian Mixture Model decision boundary (red dotted curve) and probability density function (contours) for K=2 clusters. The final cluster centers are shown as white points with an X.

on trial-and-error, and was the best combination I found to recover two distinct clusters. Perhaps a slightly better method would have been to choose ϵ = the width of the radius valley based on a k-nearest neighbor estimate, I just thought of this idea and didn't have time to implement it. The final clusters are shown in Figure 4.

2.4. *Random Forest*

I also tried one supervised machine learning algorithm, sklearn's RandomForestClassifier (Pedregosa et al. 2011), using the labeled data described at the beginning of section 2. Random Forest is an ensemble method which strives to improve the weaknesses of a simple Decision Tree. It does this by generating many bootstrapped data sets, and finds a decision tree with completely pure leaves for each. Then, the final model is decided by taking the average of each bootstrapped model. For this project, I chose 500 bootstrapped data sets to create the Random Forest model.

3. RESULTS

The final clusters for each model are shown in Figure 5. Notably, each algorithm outputted slightly different classifications. Qualitatively, one can tell where the machine made incorrect classification based on slope of the radius valley as predicted by Van Eylen et al. (2018) (plotted as a grey dotted line in Figure 5). I expected one distinct cluster below the SVM slope (super-Earths) and one distinct cluster above the SVM slope (sub-Neptunes). Based on this, K-means and GMM obviously produced some incorrect classification.

The accuracy of each model is presented in Table 1. Accuracy being the percent of data points that were predicted correctly based on the labels I assigned according to the methods in Section 2. Note that I only report a training accuracy for DBSCAN, since I did not split the data set into training and testing for this model. K-means cluster and GMM performed quite similarly with testing accuracy's $> 90\%$. This is a good accuracy score in general, but

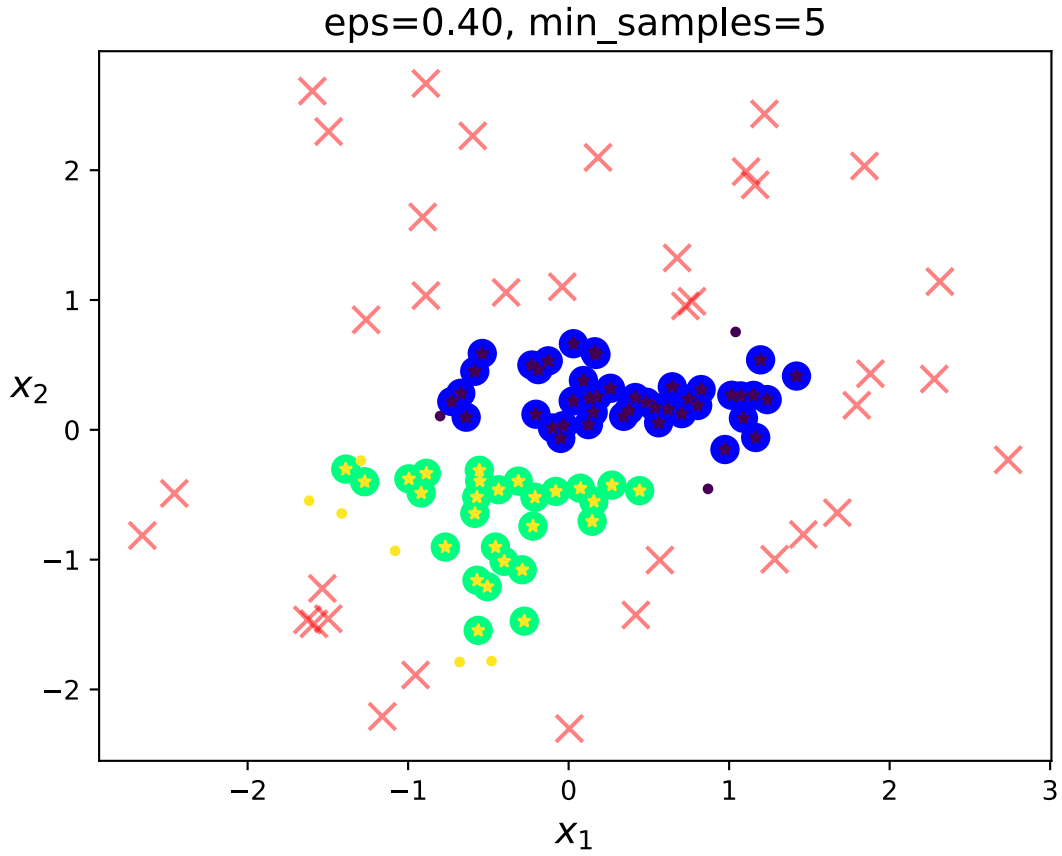

Figure 4. DBSCAN final clusters (green and blue) and noise (red X's) for $\epsilon = 0.4$ and min_samples = 10.

again Figure 5 reveals some key places where these models went wrong. The accuracy's convey that Random Forest did the best and DBSCAN did the worst at classifying the data points. However, DBSCAN has such a low accuracy because it classified a lot of data points as noise. In fact, a closer look at panel c of Figure 5 shows that besides the incorrect noise classifications, there is only one other point that is classified incorrectly. These data points were classified as noise because the ϵ value I chose is too low to encapsulate them, however switching to a larger ϵ runs the risk of obscuring the radius valley gap entirely. So, if I removed that data points that were classified as noise, and fit the model again, then I expect the accuracy of DBSCAN to be much better. Finally, the Random Forest model had 100% training and testing accuracy's, this model did a great job of classifying the data.

Algorithm	Training Accuracy	Testing Accuracy
K-means Clustering	88.4%	96.5%
Gaussian Mixture Model	89.5%	93.1%
DBSCAN	67.8%	-
Random Forest	100%	100%

Table 1. Training and testing accuracy of each algorithm. Training and testing refer to which subset of the data being evaluated.

4. DISCUSSION

The clustering algorithms only did an okay job of classifying small exoplanets. Additionally SVM certainly does a better job at this problem. I will now explore some ways that this project could be improved to possibly be more useful.

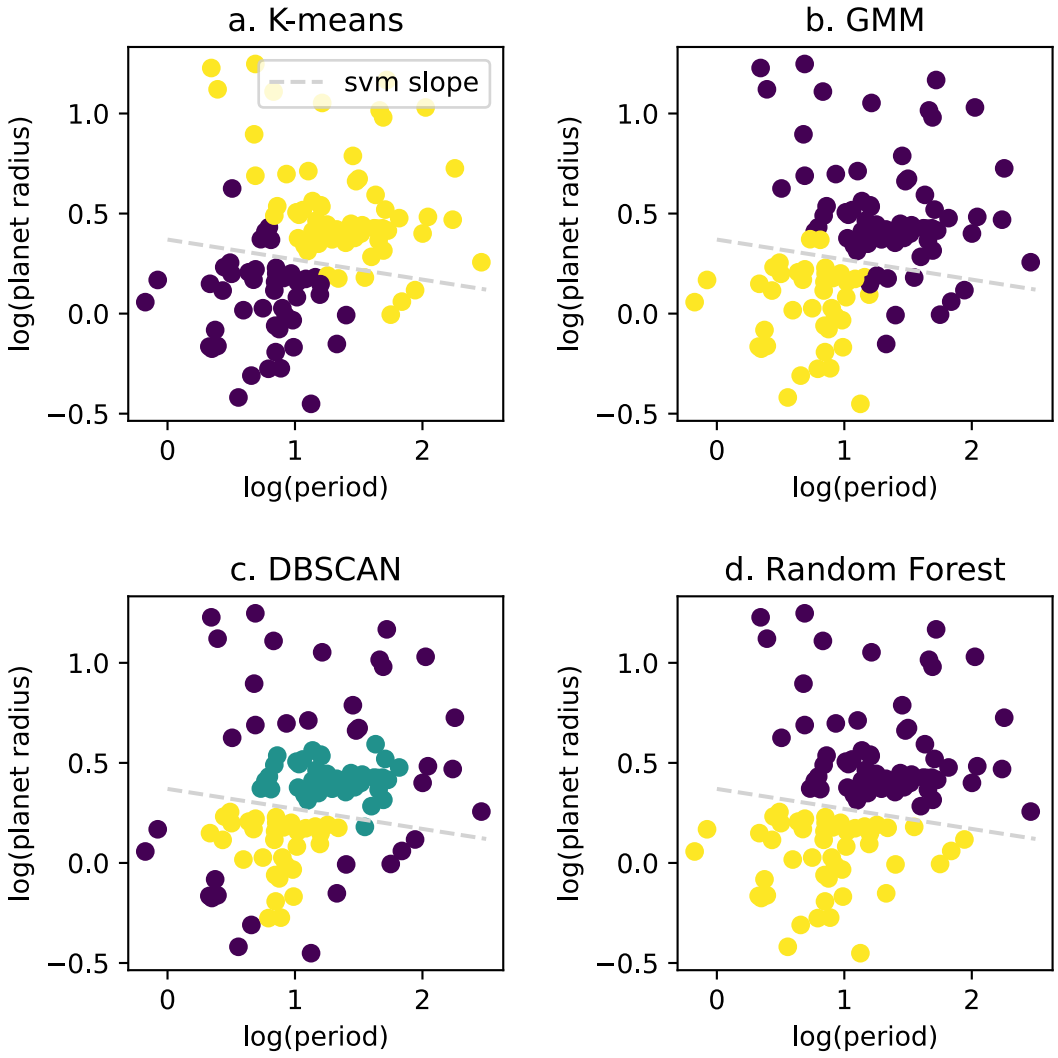

Figure 5. The final clusters for the entire data set from each machine learning algorithm. The distinct clusters are represented by different colors. In each panel, the slope of the radius valley as determined by SVM (Van Eylen et al. 2018) is plotted as a reference. Panel a: K-means Clustering. Panel b: GMM. Panel c: DBSCAN, note that points in purple were labeled as "noise" by the model. Panel d: Random Forest.

Firstly, it would be interesting to see what happens if additional features are incorporated into the models. Specifically, stellar mass could be an interesting feature to add since there has been some evidence that the slope of the radius valley may be dependent on stellar mass (Gupta et al. 2022). This could be more useful than the current methodology, since machine learning could possibly pick up on patterns that I can't see by eye in a higher dimensional space. There are other features that could be added too: stellar radius, stellar metallicity, etc.

Another improvement could be to use a larger training data set. My training data set only had 115 object. For example, Fulton & Petigura (2018) report a sample size 1334 objects that would be useful for this problem. Additionally, one could just pull all the planets from the NASA Exoplanet Archive with transit measurements. In this case, it would be smart to limit $R_P < 4R\oplus$ and $P < 50$ days.

Finally, I think it would have been helpful to incorporate bagging when training the K-means and GMM models. By generating many bootstrapped data sets, similar to how Random Forest works, I could have reduced bias during the training stage. It's possible that reducing biases could help improve performance.

5. CONCLUSION

In this project I aimed to use clustering algorithms and random forest to classify small exoplanets in two distinct groups; super-Earths and sub-Neptunes. My clustering models did not perform very well compared to past studies which used a simple SVM approach. The Random Forest model did a great job of classifying the planets with 100% accuracy. But it is still less useful than SVM because I could not recover an accurate slope of the radius valley based on the results. There are some ways in which my methods could be improved, but I suspect that unsupervised machine learning over complicates this probably and is not really necessary.

Facilities:

Software: Scikit-learn (Pedregosa et al. 2011)

REFERENCES

Borucki, W. J., Koch, D., Basri, G., et al. 2010, Science, 327, 977, doi: 10.1126/science.1185402

Dressing, C. D., Charbonneau, D., Dumusque, X., et al. 2015, ApJ, 800, 135, doi: 10.1088/0004-637X/800/2/135

Fulton, B. J., & Petigura, E. A. 2018, AJ, 156, 264, doi: 10.3847/1538-3881/aae828

Fulton, B. J., Petigura, E. A., Howard, A. W., et al. 2017, AJ, 154, 109, doi: 10.3847/1538-3881/aa80eb

Gupta, A., Nicholson, L., & Schlichting, H. E. 2022, MNRAS, 516, 4585, doi: 10.1093/mnras/stac2488

Gupta, A., & Schlichting, H. 2022, in Bulletin of the American Astronomical Society, Vol. 54, 103.02

Gupta, A., & Schlichting, H. E. 2019, MNRAS, 487, 24, doi: 10.1093/mnras/stz1230

Jontof-Hutter, D., Ford, E. B., Rowe, J. F., et al. 2016, ApJ, 820, 39, doi: 10.3847/0004-637X/820/1/39

Lopez, E. D., & Fortney, J. J. 2014, ApJ, 792, 1, doi: 10.1088/0004-637X/792/1/1

Owen, J. E., & Wu, Y. 2017, ApJ, 847, 29, doi: 10.3847/1538-4357/aa890a

Pedregosa, F., Varoquaux, G., Gramfort, A., et al. 2011, Journal of Machine Learning Research, 12, 2825

Rogers, L. A. 2015, ApJ, 801, 41, doi: 10.1088/0004-637X/801/1/41

Van Eylen, V., Agentoft, C., Lundkvist, M. S., et al. 2018, MNRAS, 479, 4786, doi: 10.1093/mnras/sty1783