Planet Hunter

Group G1(b)

Richard Aguilar

Problem description

This project will involve the use of a machine learning to detect the existence of planets around other star systems, and to cluster those planet-hosting stars using their specific stellar characteristics. The hunt for extrasolar planets, or exoplanets, is a relatively new field of Astronomy. The first such planet was discovered in 1992. Since then, as of November 1st, 2020, over 4,300 exoplanets have been confirmed in over 3,200-star systems, with a little over 25% of those star systems being multi-planetary.

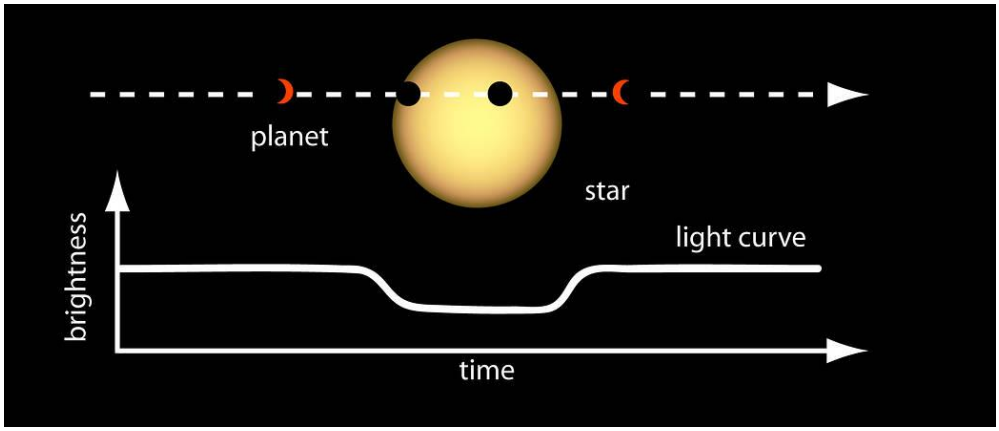
Detecting extrasolar planets is a difficult feat. These are objects that are quite literally outshone by their host star, and are relatively small in terms of size and mass. Thus, multiple methods have been developed to detect them. Three such examples are the radial velocity method, the microlensing method, and the transit method.

The radial velocity method hinges on Newton's laws of motion, specifically his third law: every action has an equal and opposite reaction. In this case, just as a star tugs on the planets orbiting it, so too does a planet tug on its star. The natural consequence of Newton's laws is that every pair of orbiting bodies in the universe revolve around a common center of mass: the barycenter. For bodies in which one is significantly more massive than the other, the common center of mass is physically inside the larger one. As such, what should happen is that the larger mass should appear to "wobble" as it revolves around the barycenter inside of it. This provides a useful way for detecting if a star is being orbited by a mass of significant magnitude—a planet.

The microlensing method is dependent on Einstein's General Theory of Relativity. His theory is a theory of gravity, providing an exclamation as to what "causes" gravity. Einstein posits that the gravity that we feel is not a "real" fundamental force, but, is in fact, a consequence of the

geometry of spacetime. Any object with mass curves spacetime, distorting it. As such, when objects traveling in a straight line happen to stumble upon a sufficiently deep "well" caused by another body, if they are not traveling with sufficient velocity, they are caught within it, and travel in a geodesic (a straight line on a curved plane). This is what we interpret as gravity, and what we see as an "orbit". Since light is affected by this curvature, if an object with sufficient mass orbits a star, it should cause light emanating from the star to bend around it, distorting the image of the star. This provides yet another useful way for detecting planets.

The third method, and the one this project will use, is called the transit method. This one is much simpler than the previous two and does not involve complicated physics. This takes advantage of the shadows cast on the host star from the smaller body. Every time a planet passes in front of a star, it causes a small dip in the apparent magnitude of the star. The amount of starlight blocked, and for how long it is blocked, is dependent on the size of the planet and its orbital period. If the star's light is blocked at regular intervals, it is likely that it is being orbited by a planet. An example is shown in the picture below:



Goals

This project makes the use of two different machine learning algorithms: the supervised machine learning model, Support Vector Machines, for the binary classification, and the unsupervised model, K-Means, for the clustering of the stars.

Team Member Roles

This project was a solo one, with myself as the researcher, programmer, and evaluator. I implemented Support Vector Machines using the Scikit-learn library, and the second, K-Means Clustering, from scratch.

Approach/Evaluation

This project makes the use of two different machine learning algorithms: the supervised machine learning model, Support Vector Machines, for the binary classification, and the unsupervised model, K-Means, for the clustering.

The first algorithm, Support Vector Machines, can be used for both classification and regression, but is most often used for classification problem. The logic behind it is simple. Find a hyperplane that best segregates the different classes in the data. In the context of two-dimensional data, that hyperplane is a line, for 3D vector spaces, it is a 2D hyperplane, and so forth. The most optimal hyperplane maximizes the distance from itself (referred to as the decision boundary), to the nearest datapoints—called the support vectors—so that samples can be correctly classified. The distance between the decision boundary and the support vectors is referred to as the “margin”, and when it comes to SVMs, the type of margins that are used depend on whether the data is linearly separable, linearly non-separable (the data has outliers), or non-linear data. In this project three different types of SVM were used, a Linear version, and two kernelized versions: Polynomial, and Radial Basis Function.

Soft-margin Linear SVM

It was first assumed that the samples would be linearly-separable, but with outliers. Thus a linear SVM with a soft-margin would be appropriate. Grid Search Cross-validation was used for the hyperparameter tuning. In this case, the penalty parameter, C, is being tuned.

Table 1: Soft-margin SVM performance

	C = 1	C = 11	C = 21	C = 31	C = 41	C = 51	C = 61
Accuracy	82.9%	83.6%	83.7%	83.8%	83.9%	84.0%	84.0%

This was done to find the optimal value. The higher the C, the more accurate it became, with its performance hitting its highest value with C = 55. The end result was the linear SVM performing adequately, with a maximum accuracy score of 81.94%, and an accompanying F1 score of 70.31%. Given that the performance was merely satisfactory, and not excellent, it is likely that using a kernelized SVM would give better results.

Kernelized SVM

The first kernel tried was the polynomial kernel. This uses a polynomial function of degree *d* to elevate the features to a higher-dimensional space. In order to find the optimal value of C and *d*, Grid Search Cross-validation was also used. The results returned in optimal value of C = 59, and *d* = 3. The table below shows the performance of the polynomial kernel from degrees 1 to 5

Table 2: Kernelized SVM performance (Polynomial, C = 59)

	d = 1	d = 2	d = 3	d = 4	d = 5
Accuracy	83.2%	82.9%	83.4%	80.8%	76.8%

The polynomial kernelized SVM had a similar performance to the linear SVM. Its accuracy was 83.67% with an F1 score of 76.38%. The last kernel function to be tested would be Radial Basis Function (RBF)

The RBF Support Vector Machine is also used to handle non-linear data like the polynomial kernel function. This uses a multi-dimensional function whose value depends on the distance between two points. This generates a 'sphere of influence' from the support vectors which then affects are how future samples are classified. The two hyperparameters here are C and gamma. The gamma term determines the magnitude of the sphere of influence. If the gamma is low, the "sphere" of influence of that support vector will be significant, whereas if it's high, the sphere of influence will be smaller. Using Grid Search Cross-validation, the optimal values for C and gamma were 100,000,000 and 0.00001, respectively. This resulted in the Kernelized RBF Support Vector Machine achieving an accuracy of 83.43%, and an F1 score of 73.90%

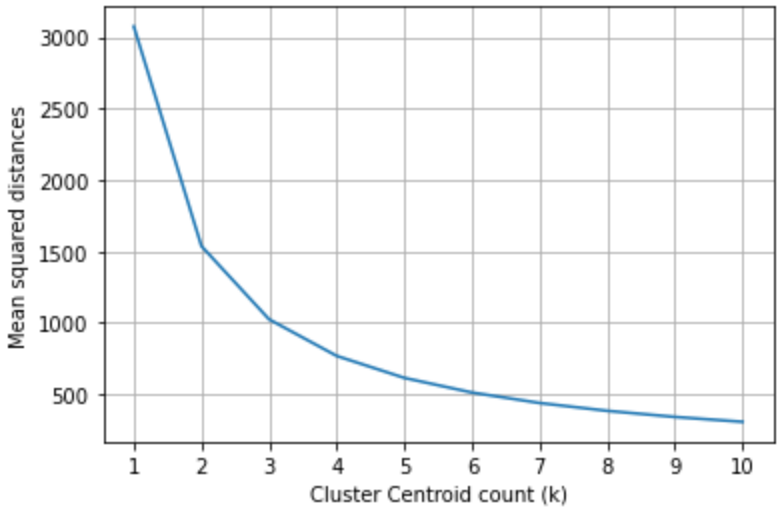
Table 3: Kernalized SVM performance (Radial Basis Function, C = $1.0\ x\ 10^{8}$

	$\gamma = 1.0\ x\ 10^{-5}$	$\gamma = 1.0\ x\ 10^{-6}$	$\gamma = 1.0\ x\ 10^{-7}$
Accuracy	83.1%	82.1%	80.8%

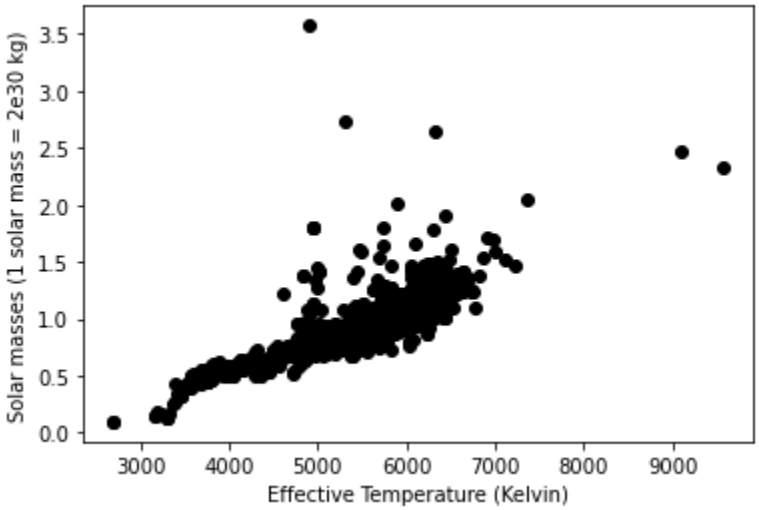
K-Means Clustering

This is an unsupervised machine learning algorithm used for clustering problems. K-Means is a relatively simple model that works by clustering dataset into a pre-defined number of groups, “k”. It does so by initially plotting cluster seeds, and then grouping the data based on how close it is to the centroid using a distance metric, such as a squared Euclidean distance. It then takes the means of each cluster, and iteratively selects new centroids by placing them on the means of the clusters in the last step. This process repeats, until a maximum number of iterations have been reached, or until there is very little to no change in the percent difference between the mean centroids.

In order for this algorithm to work most optimally, it’s hyperparameter, “k”, needs to be tuned. There are many different methods for finding the most optimal k, but the one used here was the “Elbow Method”. The Elbow Method works by calculating the mean squared distance between the samples, and their respective centroids. In my implementation of the Elbow Method, I created an array of k’s, for which the algorithm would iterate through. Each value of k would then also be run five times, and have the mean distances of every samples in each iteration recorded. After which, the mean of the five mean distances would be recorded. The result ended up being the typical Elbow Method graph with a inverse relationship between the mean squared distances and the number of cluster centroids. The graph is shown below:



The optimal value ended up being at k = 3. This makes sense in the context of the data, as given the range in effective temperature and masses of the stars, there are roughly three types of stars in this dataset: dwarf stars, main sequence sun-like stars, and giant stars.



Discussion/Challenges

The use of machine learning within astronomy, and within the context of discovering exoplanets, is a relatively new phenomenon. This was largely due to technical barriers, and ignorance of the

existence of exoplanets. The first one was only just confirmed in 1992, and as of December 2020, there are less than 5,000 confirmed planets. In addition, satellites specifically designed to search for exoplanets are also new, with the Kepler Space Telescope being the first. Thus, this the use of machine learning to comb through hundreds of thousands of samples of light-curve data is a novel one. It is clear, however, that more sophisticated machine learning models will be necessary, since the accuracy was relatively low for the support vector machines algorithm. Although it achieved an accuracy of ~84%, that is not very high compared to other uses of this same algorithm in binary classification.

Two of the major challenges and goals I did not achieve were implementing the support vector machine algorithm from scratch, and the inability to visualize the results from either algorithm on a 2D graph. The mathematics behind support vector machines were significantly more complicated than I had anticipated. While a fair amount was understandable to me, the optimization portions were quite confusing. I have never taken a mathematical optimization course, so the aspects associated with it, such as the use of “Lagrange multipliers” unfortunately went over my head.

When it came to visualization, this was also something I could not achieve. The dataset from NASA’s exoplanet archive is multidimensional data containing more than 80 features. I managed to reduce the information I needed down to less than 20. Given that this is data with dimensions greater than 3, I had to figure out a way to reduce it down to either 2 or 3 dimensions, so that the visualization could be understood. I understand that there are two ways to do this: Principal Component Analysis (PCA), and Linear Discriminant Analysis (LDA). Unfortunately, I couldn’t figure out how to implement the logic in python, therefore I was not able to visualize the data.

Given that one of the methods is reliant fairly heavily on visualization (K-Means clustering), this is rather disappointing.

References

Ansdell, M., Ioannou, Y., Osborn, H. P., Sasdelli, M., Smith, J. C., Caldwell, D., . . . Angerhausen, D. (2018). Scientific Domain Knowledge Improves Exoplanet Transit Classification with Deep Learning. *The Astrophysical Journal Letters*.

Armstrong, D. J., Gamper, J., & Damoulas, T. (2020). Exoplanet Validation with Machine Learning: 50 new validated Kepler planets. *Monthly Notices of the Royal Astronomical Society*.

Sturrock, G. C., Manry, B., & Rafiqi, S. (2019). Machine Learning Pipeline for Exoplanet Classification. *SMU Data Science Review*, Vol 2: No.1, Article 9.