Searching for Correlations Between Lunar Crater Features and Location with Machine Learning

Braden Oh* and Michael Remley†
Olin College of Engineering, Needham, MA, 02492

Modern machine learning algorithms may be employed to search for correlations between the properties and location of lunar craters. This study investigates the possibility of using machine learning classifiers to identify new patterns that legacy data could not reveal by training classifiers on a database released by Dr. Stuart Robbins in 2018, the most complete lunar database to date. Two types of classifiers, random forest and multi-layer perceptron, were each trained on twin datasets of craters grouped by location, using either USGS quadrangle or k-means clustering. Following training, the models were presented with a set of craters not present in the training data for which all data in the Robbins database except for latitude and longitude were provided. Both classifiers performed significantly better at classifying craters than random guessing when using quadrangle-grouped data; however, both classifiers placed the highest weight on the longitudinal error in ellipse fits, which is correlated with latitude. As a result, we conclude that machine learning classifiers are not able to easily uncover patterns between crater geology and lunar location due to the strongest predictors of crater location not being intrinsic features of the craters themselves.

I. Introduction

One of the great stories humanity seeks is that of its creation. This inquiry about the origin of our universe, our planet, and ourselves is augmented by studying the formation of the Moon. Solar system formation is a chaotic process, meaning it is difficult to reconstruct the history of planetary formation only from physical laws and known conditions [1, 2] . Unable to extrapolate, we directly study planetary formation and assume that the same processes which formed one body in the solar system were also formed the others [2]. With this assumption, our study of the Moon provides insights into various planetary formation processes, including those that cause: surface feature formation [3]; variations of isotopes observed in geology [4]; results of enormous planetesimal impacts [5]; and dry landslides on Mars [6]. This helps us build a chronological story of events in the early solar system [7] and allows us to reconstruct our story of creation. This narrative is scrawled (at least in part) upon the surface of the Moon - further investigation of which provides clues to our origin story.

Robbins [8] accompanied the release of the Robbins Lunar Crater Database to describe the methodology, limitations, and possibilities of the most complete lunar database to date. Crater data were derived from manually selected rim points on images from the Lunar Reconnaissance Orbiter (LRO) Camera's wide-angle camera (WAC), the LRO's Lunar Laser Altimeter (LOLA), and the Terrain Camera (TC) on SELENE/Kaguya. The database contains position, orientation, circle fits, ellipse fits, fit quality, and the number of points used to identify each crater. Multiple sources of random variation between 1 and 10% are present from projection errors and human variation. Robbins' ellipticity and spatial density analyses are consistent with previous databases, and new trends were not immediately apparent. As perhaps the most complete database to date, Robbins emphasizes possibilities for future work to spot patterns where legacy data could not reveal any. We ask the question of whether machine learning algorithms can identify patterns between crater features and location on the moon, seeking to uncover some of these patterns into which Robbins emphasized future research.

*Engineering Physics, class of 2023
†Electromechanical Engineering, class of 2022

II. Methods

A. Data Pre-Processing

We took the database from [8] and performed three pre-processing steps before using the data to train a pair of machine learning models. We began by removing any craters containing a *NaN* (not a number) value. This required us to remove eight craters; being that the entire database contains 83,061 craters, the removed values compose less than 0.01% of the database. Next, we grouped the craters into thirty regions (a detailed description of this process and the reasoning behind it is given in Section II.B). Each crater was assigned to a region represented by an integer ranging from 0 to 29, inclusive. This region index was added to each crater in the database, represented as a new feature column. Our final pre-processing step was to remove exact location information and irrelevant data from the database. We removed six pieces of information from each crater: 'Unnamed: 0', 'CRATER_ID', 'LAT_CIRC_IMG', 'LON_CIRC_IMG', 'LAT_ELLI_IMG', and 'LON_ELLI_IMG.' The first two columns removed are index columns that provide no information about the crater features, so serve no purpose in the machine learning training process. The latter four columns contain the latitude and longitude coordinates for the centers of a circle and ellipse fit to each crater. These location coordinates are removed because a crater's location is sufficiently captured by its assigned group and because allowing exact coordinates to remain would serve as a "dead giveaway" to a machine learning algorithm seeking to predict crater location by features alone.

B. Region Assignment

We used two region assignment methods: the United States Geological Service (USGS) lunar quadrangle system and k-means clustering. Each crater is labelled according to the region in which it lies so that a machine learning algorithm can predict the label again later given solely non-positional data. Both of these region labelling methods add a new column to the lunar crater database called "LABEL" which stores an integer index for each crater region.

1. USGS Quadrangles

The USGS has divided the moon into thirty quadrangles at the 1:2.5 million scale. The division gives regions that are more equal in area than if the latitude and longitude slices were all equal intervals. The USGS labels these Lunar Quadrangles (LQ) regions from 1-30 as "LQ01" to "LQ30", but we have labelled them with integers 0-29 where 0 maps to LQ01, 1 to LQ02, and so on, because computers index from 0. Figure 1 depicts this subdivision on a latitude vs. longitude graph of lunar crater positions. Observe that certain regions, such as 10, are less dense than others. This uneven distribution is quantified in the corresponding histogram in Fig.1.

2. k-Means Clusters

k-means clustering is an algorithm that seeks to group data into k clusters. This results in the distribution of craters in each region being more even than in the quadrangle grouping. The k-means algorithm was run on 'LAT_CIRC_IMG' and 'LON_CIRC_IMG', the latitude and longitude coordinates of the center of each fitted circle for $k = 30$, that is, to cluster the data into 30 regions.

The clustering algorithm begins by selecting k random points to use as cluster centroids. The clustering algorithm then alternates between two steps: *assignment* and *update*. During the assignment step, each crater is assigned to the cluster with the nearest centroid. During the update step, the mean of each cluster is calculated and the centroid is moved to that mean location. The algorithm then re-assigns the craters to these new centroids and the algorithm repeats until the craters are consistently re-assigned to the same cluster. Because the initial points are randomly selected, the algorithm is not guaranteed to find the optimal solution, just a converged solution. For this reason the craters are not equally grouped in each region, but are significantly better distributed than in the quadrangle regions.

To select the number of clusters we would use, we swept a range from 15 to 45 clusters, evaluating a single decision tree classifier (a detailed description of decision tree classification is provided in section II.C.2). We discovered that decreasing the number of clusters (and thus reducing the number of labels for a classifier to select from) increased the accuracy of the decision tree but also increased the probability of success by random guessing. Discerning a negative correlation between accuracy and number of clusters, we decided to use 30 clusters so that we could easily compare the results from training on the k-means clustered craters to the results from training on the quadrangle-grouped craters.

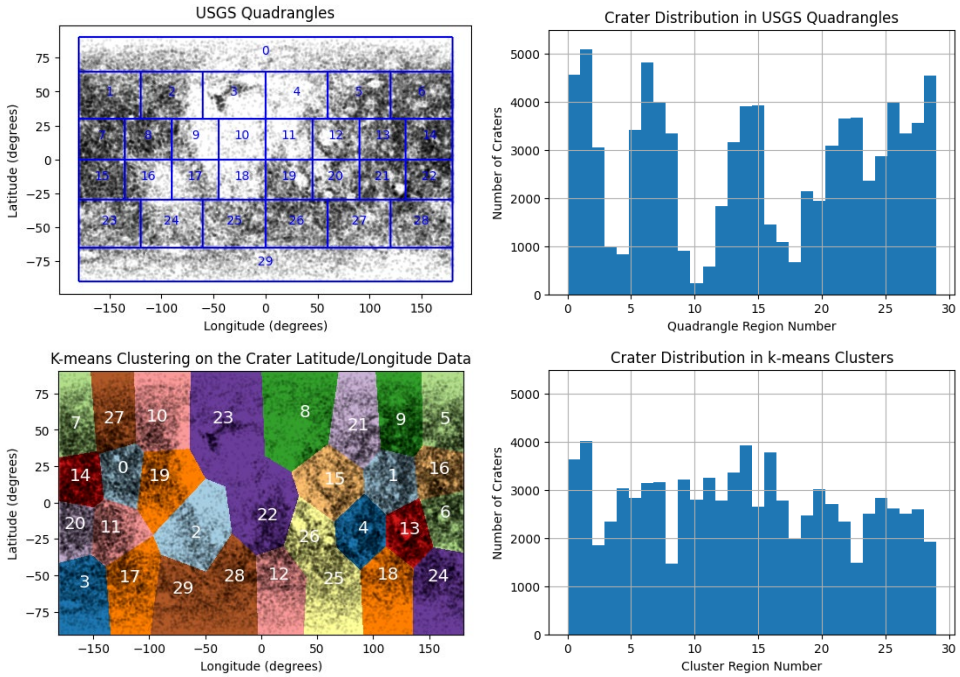

Fig. 1 (Top left) Map of lunar crater positions with USGS quadrangle overlay. Crater centers plotted with 10% transparency. (Top right) Distribution of lunar craters in USGS quadrangle regions. This distribution could bias the algorithms to classify craters into certain regions simply because they have more craters. If this bias is present, regrouping our data more evenly would significantly change the error rate. (Bottom left) Map of lunar crater positions regrouped by k-means clustering. Crater centers also plotted with 10% transparency. (Bottom right) Distribution of lunar craters in 30 k-means regions. These regions are more evenly distributed and consistently produce higher error rates than the USGS regions.

C. Classifiers

We tested two different *classifiers*, algorithms which take some input data and sorts them by group. Both of our classifiers take the 16 non-positional data columns as input and classify the crater into one of 30 regions. A randomly guessing classifier would guess correctly a mere 3.33% ($1/30^{th}$) of the time. Since guessing correctly is incentivized, unequal distributions present a problem where accuracy is increased by guessing regions that have more craters. Two ways of handling this are to train classifiers on an equal-distribution subset of craters or to have equally-distributed craters to begin with. The USGS quadrangles distribution presents the unequal distribution problem, so we introduced k-means clustering to generate a more equally distributed dataset.

Classifiers are a powerful tool to automatically recognize patterns where the human eye may have overlooked them. If the classifiers are consistently good at recognizing some craters more than others, then the well-recognized craters must be distinct in a measurable way because this would not happen consistently by chance.

1. Data Splitting

Classifiers need to be tested on data which is not included in the training set. The training data and testing data must have no overlap but should be statistically similar. To determine what percentage of data we would reserve for testing, we experimented on single decision tree classifiers at a time (a detailed description of decision tree classification is provided in section II.C.2), sweeping a range from 1% to 40% of data reserved for training. We discovered that reducing the data provided for training made little difference until around 25%, when the accuracy of the decision tree began to gradually decline. We concluded that little advantage could be gained by reducing the training set and so decided to use

99% of the data (82222 craters) for training and reserved only 1% (831 craters) for testing.

2. Decision Trees and the Random Forest

The decision tree is a classifier that makes a series of binary decisions in order to predict a target value for a data point. In the context of craters, a training set of crater features along with a list of correct location labels is fed into a decision tree model, and the model builds a 'tree' of if-then statements that begins with a single switch point and ends with a location label. For an unknown crater, the first switch is checked and then the data point is passed to a subsequent switch and so forth until finally reaching a label. The various crater features are weighted by importance and appear in the switch hierarchy accordingly.

One primary advantage of the decision tree over other machine learning algorithms is that it can be easily understood. The tree can be conceptualized in a simple way and the importance of particular attributes is directly reflected in the hierarchy of switch statements. Another advantage is that the tree is robust to input data, requiring no normalization or scaling of the input data prior to training. A primary disadvantage of the decision tree is that it is highly sensitive to changes in the training data. Small changes in training data can result in large changes to the final tree, meaning two randomly extracted test sets may result in two classifiers that differ in accuracy by entire percentage points. Furthermore, the training process is susceptible to over-fitting the data, resulting in decision trees that are highly complex without being more accurate.

One way to combat the limitations of the decision tree is to train an ensemble of decision trees and use the output of each tree in a vote to produce a final classification. Individual trees are trained on subsets of the training data, so vary in their decision processes and accuracies. This is known as a *random forest* and is effective at resisting the tendency of trees to over-fit the data. We found that training a forest of 100-200 trees led to reliably better classifications than were produced a single decision tree, but training such a forest required significant computational resources.

3. Multi-Layer Perceptron

A perceptron is a simple neural network with an input layer with one neuron for every input and an output layer with a neuron for every output. Every input neuron is connected to every output neuron, and the weight of each connection determines how much activation at the input neuron is forwarded to the output. A multi-layer perceptron (MLP) introduces one or more hidden layers between the input and output layers to allow the network to learn complex behaviors. In either case, the entire network starts with random-valued weights and is trained on a set of data by incentivizing weights that tend to produce the correct output. In our case, we have 16 input neurons for each of the 16 data columns, 20 neurons chosen in a single hidden layer, and 30 neurons in the output layer for each of the 30 regions. Whichever output neuron activates the most indicates the region the perceptron thinks the column came from.

We present two perceptrons, one for each type of region classification. Both have the same structure and are trained and tested in the same way (10% of data used for testing). Unlike the random forest classifier, the MLP classifiers require data normalization to fall within a range of 0 to 1 within every column in the database. The final relative scaling of the weights on the input layer indicate which data columns are most important in determining a crater's region, and some guesses can be made about combinations of parameters by examining the hidden layer. Features that are important to both the perceptrons and random forests may reveal new patterns in the data.

III. Results

A. The Confusion Matrix

A convenient way of gaining insights into where a classifier makes errors is via a visual representation known as a *confusion matrix*. A confusion matrix is an $n \times n$ table with rows that represent the true label of a data point and columns that represent the model's predicted label. Each cell contains an integer which represents the number of craters which the model classified in that way. Scikit Learn [9, 10] provides a built-in method for plotting confusion matrices which represents the predicted labels in ascending order along the x-axis and the true labels in descending order along the y-axis. For each cell, the number inside represents the number of craters with an actual label, y, to which the classifier attached a predicted label, x. Thus, in this representation, the cells which correspond to correctly predicted craters fall along the main diagonal which runs from top left to bottom right. Figure 2 shows the confusion matrix for a highly accurate random forest. The model which produced that matrix was trained on a dataset that included the true labels, so the high accuracy is no surprise.

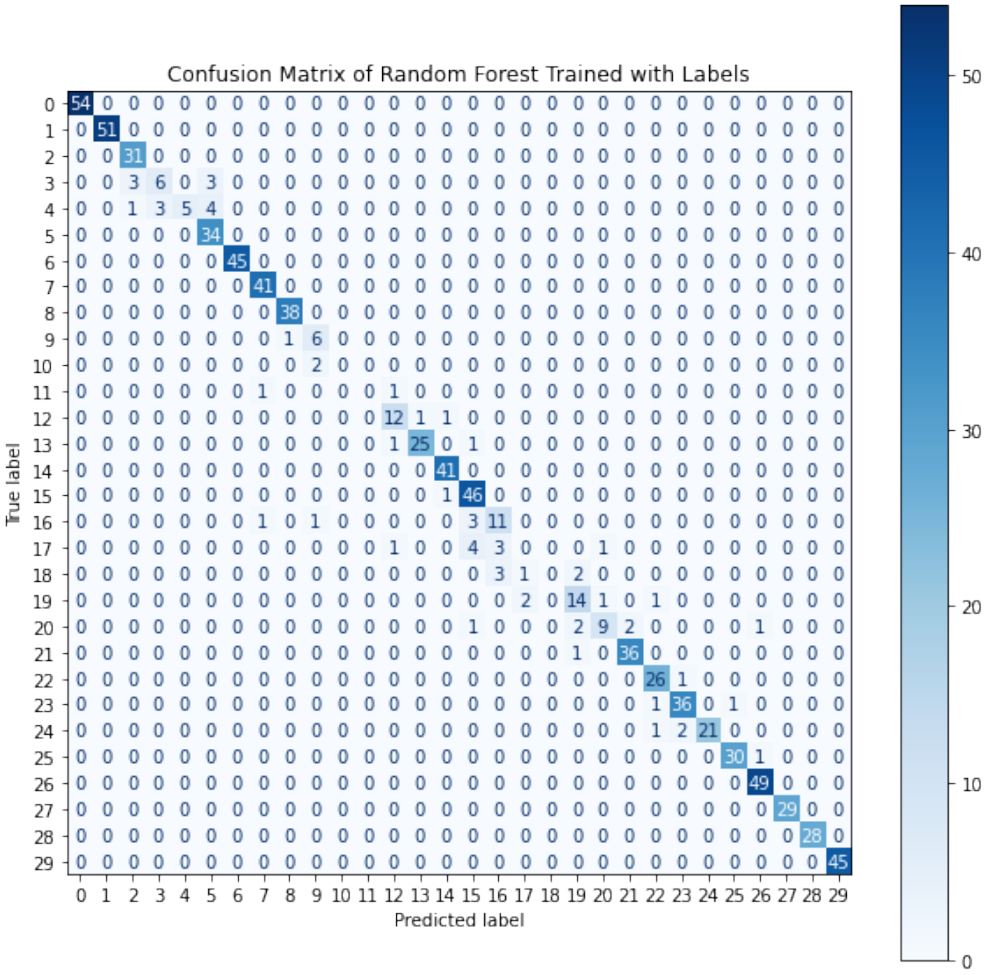

Fig. 2 The confusion matrix from a random forest trained with the location labels present in the training set. This forest has an accuracy of 92.5% at identifying craters in the test set and the high accuracy is visible in the diagonal line across the matrix. The Y-axis of the confusion matrix shows the true location of each crater while the X-axis shows the location predicted by the random forest. Each integer label corresponds to a USGS quadrangle.

B. Decision Trees and the Random Forest

We began by training and testing a variety of single decision trees on data grouped by k-means clustering and found accuracies that generally fell within the range 7.5-9%. We then performed an identical set of tests on data grouped by quadrangle and discovered that the accuracies averaged significantly higher, generally falling within 11-13% accurate. Repeated testing proved this disparity between groupings to be consistent, so we decided to investigate further as we moved ahead to training a random forest.

Training random forests proved to be highly computationally intensive for large numbers of trees, so we swept a range of tree numbers to find out whether accuracy was truly a function of forest size. We trained forests that ranged from 50 to 200 trees on data grouped by both quadrangle and clustering. We discovered that for forests larger than 100 trees, accuracy increased with additional trees, but the gains were marginal against rapidly increasing training time and model size (a forest of 200 trees took nearly 3 minutes to train and produced a model that consumed over 4GB of memory). As in previous tests, grouping the data by quadrangle produced a higher accuracy than by clustering, but the disparity was far larger in a random forest: clustered grouping consistently produced results over 10% less accurate.

Ultimately we decided to settle on 100 trees for our forest size, requiring a reasonable amount of training time and disk space. We trained 25 forests of this size on the data grouped by USGS quadrangle, with identical training and

testing sets, and recorded the accuracy of each. These forests had accuracies that ranged from a minimum of 16.25% to a maximum of 18.29%, with a mean of 17.09% and a standard deviation of 0.65%. Thus, these random forests performed, on average, 5.13 times better than random guessing. After gathering these data we continued to train individual forests with randomly extracted training sets to find out whether one could exceed 18.29% accuracy. Ultimately, we managed to train a forest with 19.25% accuracy, from which we extracted the importance of each feature in the training set and produced a confusion matrix. These two graphs are shown in Fig. 3.

The feature weighting graph provides insight into which features included by Robbins proved most significant in classifying the craters. The feature of greatest importance is 'LON_ELLI_SD_IMG,' carrying 26.80% of the weight of classification. This parameter is described by [8] as the "Formal standard error in the ellipse fit's center longitude, in degrees." This error increases in part because an slight longitudinal distance error in distance units will produce a higher longitudinal error in degrees near the poles. As a result, this column is a proxy for latitude since it would be highest at high latitudes, even if the distance error is constant for all craters. The decision tree seems to have spotted this and uses it to identify polar craters.

The confusion matrix yielded a light diagonal line, indicating its successes. The matrix also provided the insights that the random forest struggled to tell apart craters in regions 0 and 29 and in regions 21 and 22. Regions 0 and 29 correspond to the poles of the moon, indicating that the random forest struggled to tell apart craters at the north and south poles, while regions 21 and 22 are adjacent regions that are fairly crater-dense. The random forest also tended to over-assign craters to regions 1 and 6. These regions (which correspond to USGS quadrangles LQ02 and LQ07, respectively) are adjacent regions at northern latitudes that are crater-dense and enclose very little mare. Furthermore, the random forest performed very well in region 16.

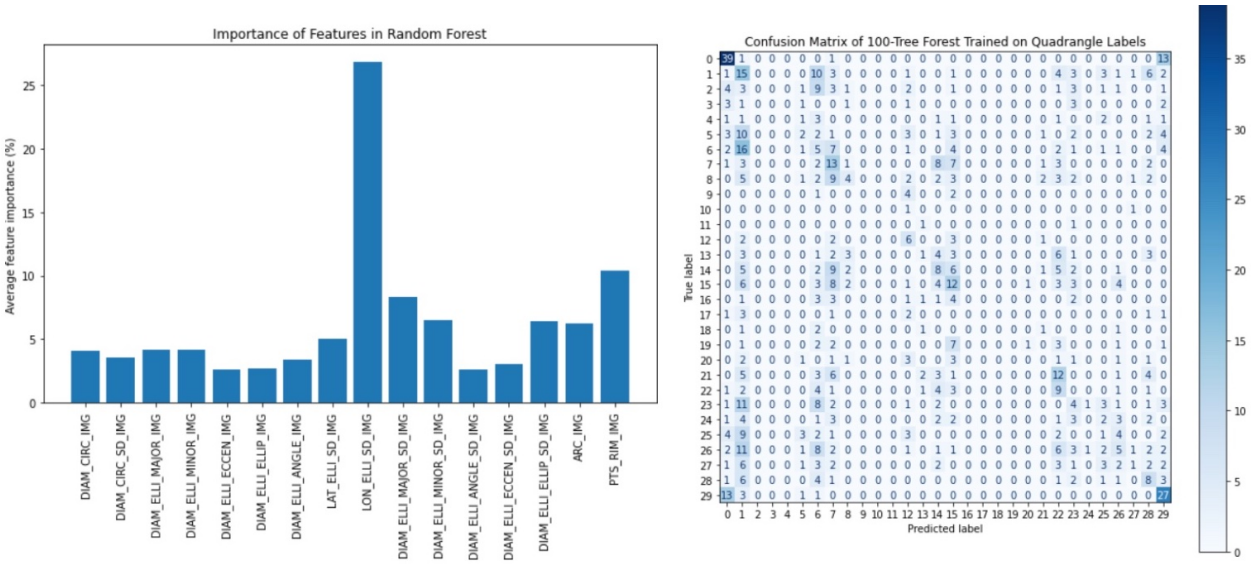

Fig. 3 (Left) The 16 crater features used for training and their relative importance in a random forest of 100 decision trees. Each tree has its own weighting of feature importance; this graph shows the average importance of each feature, taken as an arithmetic mean across all 100 trees in the forest. (Right) The confusion matrix for the same forest. This random forest has a classification accuracy of 19.25%. Higher numbers of predicted craters are represented by darker squares. The true location is shown on the y-axis and the predicted on the x-axis. A perfect prediction would result in a diagonal line from top left to bottom right; the faint diagonal line in this figure is an indicator of the algorithm's successes.

C. Multi-layer Perceptron (Neural Network)

The accuracy of the multi-layer perceptron (MLP) was around 8-14% when we first trained on both k-means and USGS regions. We discovered the MLP was bottle-necking through a 2-neuron hidden layer, significantly hindering its performance. After retraining both models, we arrived at the results depicted in Fig. 4 and Fig. 5. The accuracy of the k-means MLP was 13.6%, much worse than the 20.0% achieved by the USGS MLP but still better than the baseline

3.33% chance from random guessing. This decrease in performance is likely due to the undermined strategy of guessing the most populated regions disproportionately more. This explanation is further supported by the observation that the k-means MLP predicted all but one region at least ten times, while the USGS MLP failed to predict 9 regions a meanningful number of times. Although further from the ideal diagonal matrix, the k-means MLP may be more likely to have identified meaningful characteristics of the dataset than the USGS MLP.

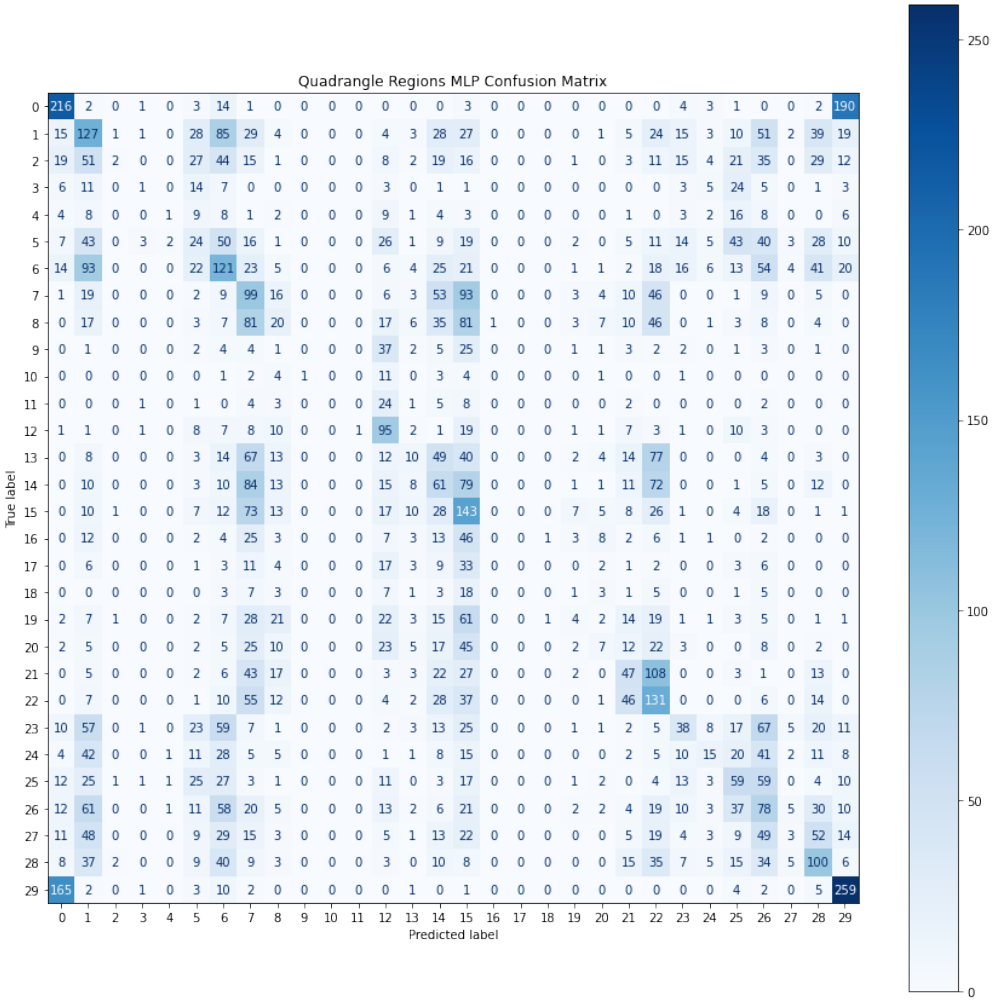

Fig. 4 The confusion matrix for the multi-layer perceptron using crater data categorized into USGS regions. Polar craters are usually correctly identified as polar, though the classifier confuses the poles nearly half of the time. Regions such as 0, 1, 15, 22, and 29 are guessed most often because they were represented the most in the training data.

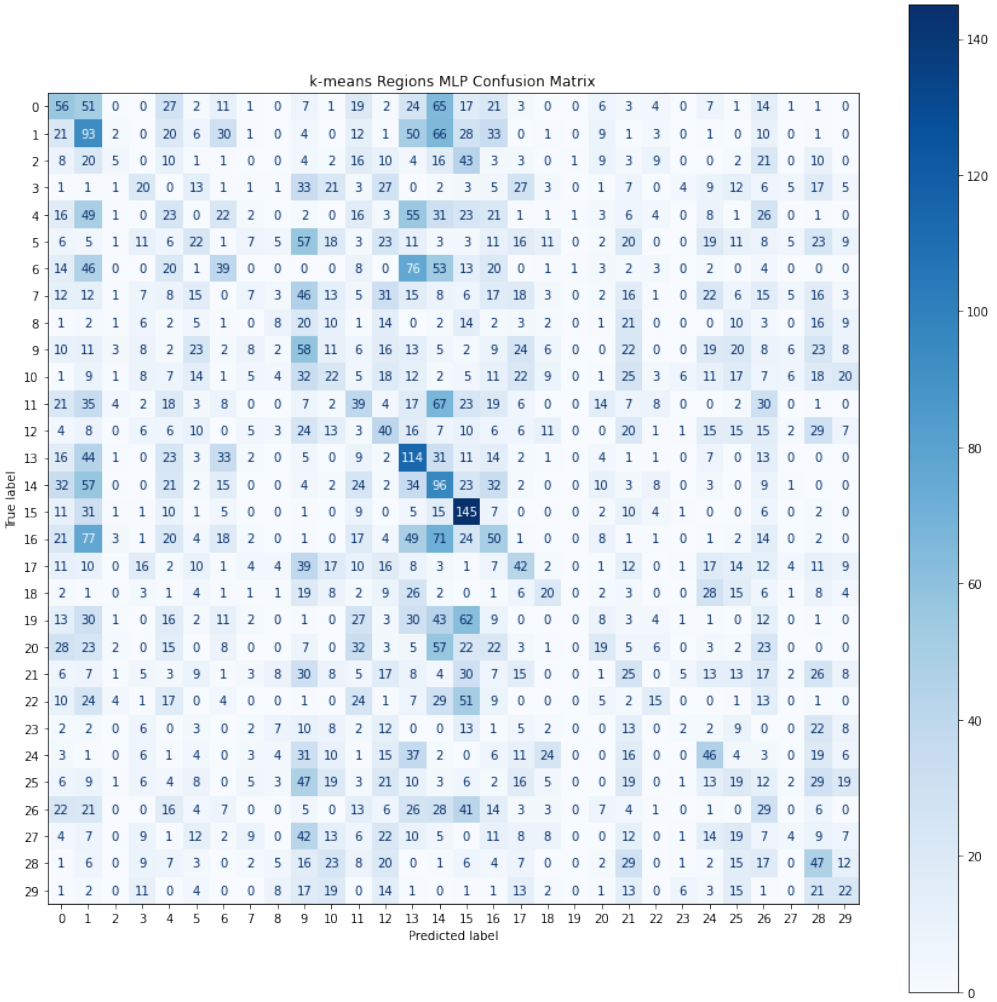

Fig. 5 The confusion matrix for the multi-layer perceptron using crater data categorized into regions found by k-means clusters. Overall accuracy is lower and the confusion is more diffuse across the whole matrix, but region 15 seems to stand out as one that is guessed most often. This region happens to be the sea of tranquility.

In Fig. 4, the corners of the matrix show that craters are accurately identified as belonging to one of the two poles, perhaps because the uncertainty in position is highest for these craters. The classifier also confuses north polar craters with south polar craters often- and vice versa. Other regions, such as region 1 (LQ02) and 15 (LQ16), were inaccurately predicted more often than the true number of craters in each, a sign that over fitting may be occuring in the USGS MLP. The results in Fig. 5 are more evenly distributed and lack the polar confusion since the poles are divided among several regions. Interestingly, 50% of craters belonging to region 15 were correctly classified. This area corresponds to the sea of tranquility.

We also depict the weights of the MLPs in Fig. 6 and Fig. 7 . From the relative sizes of the weights depicted by their thickness, we observed that in both networks, the uncertainty in position and uncertainty in the major axis of an ellipse fit of craters were the most important determinants of the region in which they would classify. The next most important parameters are related to the size of the crater, including the circular fit diameter and points used for the fit. The importance of uncertainty in longitude from the ellipse fit (LON_ELLI_SD_IMG) is consistent with the decision tree classifier. In Fig. 8, we can see that the longitude standard deviation in the ellipse fit is indeed related to whether a crater is close to the poles or not in the original lunar database. Other agreements between the two classifiers are less common; the decision tree classifier lists PTS_RIM_IMG as second-most important while neither MLP classifier does so.

In summary, the accuracy of the random forest classifier was 19.25% with quadrangle regions and 3.13% with

clustered regions. For the MLPs, the accuracy was 20.0% with quadrangle regions and 13.6% with clustered clusters. This significant performance suggests a strong correlation between crater features and location on the Moon. Patterns exist between intrinsic crater features - such as ellipticity, size, and measurement accuracy - and location on the Moon. This is an encouraging sign for machine learning's ability to uncover hidden patterns within lunar crater data and opens a door to future geologic analysis with modern machine learning tools.

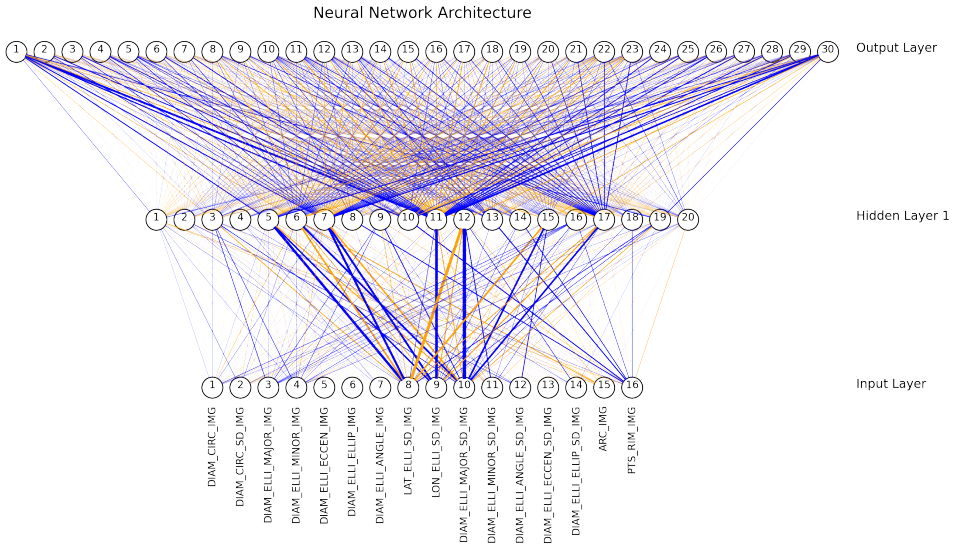

Fig. 6 The network structure for the MLP trained on USGS quadrangle data. Blue weights are positively-valued, and orange weights are negatively-valued. The size of the lines representing each weight indicate their relative weighting. From the strong weights associated with LAT_ELLI_SD_IMG, LON_ELLI_SD_IMG, and DIAM_ELLI_MAJOR_SD_IMG, we infer that these parameters are most important for classification. Created with Liu [11].

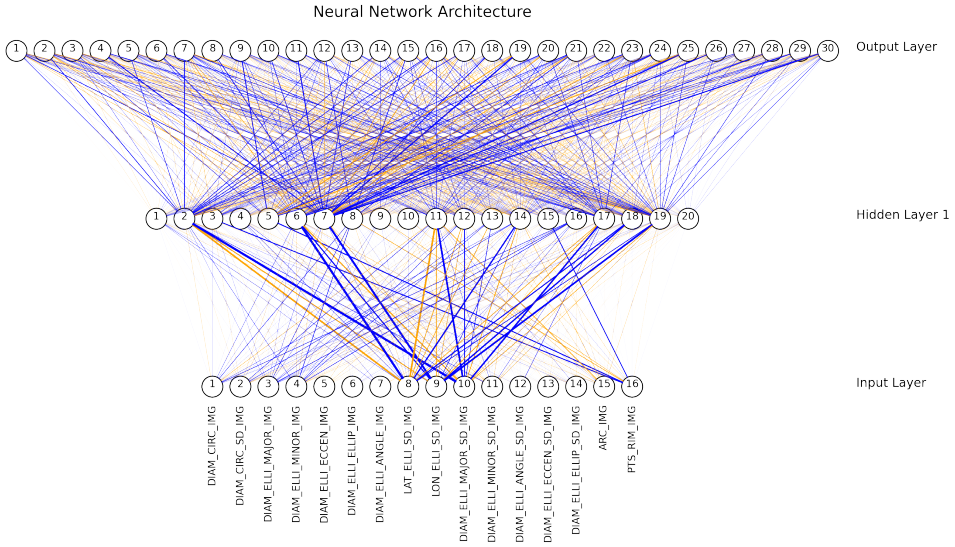

Fig. 7 The network structure for the MLP trained on k-means cluster data. Created with Liu [11].

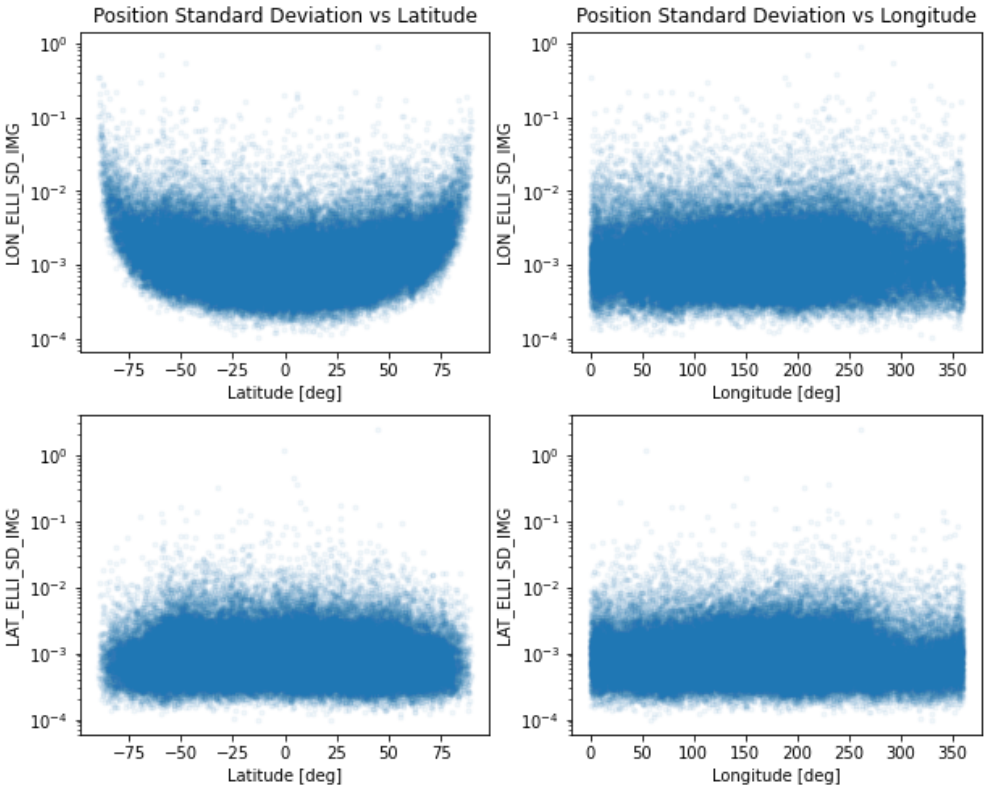

Fig. 8 Plots of position standard deviation vs position for ellipse fits. Longitudinal standard deviation increases near the poles because the degree error per unit distance at high latitude is greater than the degree error near the equator. The classifiers both identified this as a way to recognize polar craters.

IV. Conclusions

Both the random forest and the multi-layer perceptron classifiers identified the longitudinal error in ellipse fits as a strong predictor of latitude, correctly identifying many craters near the poles. Beyond this longitudinal error the classifiers diverged on the weighting of other parameters, with the random forest next favoring the number of rim points, and the perceptrons favoring ellipse fit major axis standard deviation. In conjunction with the longitudinal error as a predictor of latitude, the random forest and perceptrons both achieved higher accuracy with data grouped by USGS quadrangle as opposed to k-means cluster. We believe that this is due to the fact that both classifiers most highly weighted the database feature 'LON_ELLI_SD_IMG.' The graph of this feature against the latitude, as shown in Fig. 8, shows a strong indication that polar craters do not have low errors and indicates that latitudes very near ±90° have high errors. With the USGS quadrangles, only two regions correspond to the poles (indices 1 and 29, the north and south pole, respectively), allowing the classifiers to decide between only two regions for classification of a polar crater. With the clustered data, however, the polar craters are distributed across many more regions, making classification into a specific region more difficult for a classifier. The final accuracy of the random forest classifier was 19.25% with quadrangle regions and 3.13% with clustered regions. For the perceptrons the accuracy was 20.0% with quadrangle regions and 13.6% with clustered clusters. Both quadrangle groupings performed nearly six times better than would be achieved by random guessing. Overall, however, the strongest predictors of crater location were found to be errors in the measured position and size of craters, not intrinsic features of the craters themselves. This indicates that machine learning was not able to uncover patterns between crater geology and lunar location, so our overarching question about machine learning's potential to contribute to lunar geology and crater study remains unanswered.

Acknowledgments

This work was performed as a class project for the Spring 2020 section of the course Astronomy & Statistics. We thank Prof. Carrie Nugent for all of the effort she put into adapting this course for remote learning as a result of the the COVID-19 pandemic. We also thank the editors and reviewers of the Olin Undergraduate Research Journal for the work they put into establishing this new journal.

References

[1] Whipple, F., "The History of the Solar System," *Centennial: First Scientific Session*, Vol. 54, 1964, pp. 565–594. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC300311/pdf/pnas00182-0369.pdf.

[2] Shoemaker, E., *Interpretation of Lunar Craters*, No. Chapter 8 in Physics and Astronomy of the Moon, Academic Press Inc., 1962. URL https://books.google.com/books?hl=en&lr=&id=A883BQAAQBAJ&oi=fnd&pg=PA283&dq=lunar+craters&ots=cm8_UGrnwr&sig=llaCjlSnZ3vxQDXYIuvowi8f28A#v=onepage&q=lunar%20craters&f=false.

[3] Singer, S., "The early history of the earth—moon system," *Earth-Science Reviews*, Vol. 13, No. 2, 1977. URL https://www.sciencedirect.com/science/article/abs/pii/0012825277900216.

[4] Halliday, A., "Terrestrial accretion rates and the origin of the Moon," *Earth and Planetary Science Letters*, Vol. 176, No. 1, 2000. URL https://www.sciencedirect.com/science/article/abs/pii/S0012821X99003179.

[5] Hartmann, W., "Moon Origin: The Impact-Trigger Hypothesis," *Conference Proceedings*, 1986. URL http://adsabs.harvard.edu/pdf/1986ormo.conf..579H.

[6] Bart, G., "Comparison of small lunar landslides and martian gullies," *Icarus (Elsevier)*, Vol. 187, No. 2, 2007. URL https://www.sciencedirect.com/science/article/abs/pii/S0019103506004106.

[7] Hiesinger, H., van der Bogert, C. H., and Pasckert, J. H., "How old are young lunar craters?" *Journal of Geophysical Research*, 2012. URL https://doi.org/10.1029/2011JE003935.

[8] Robbins, S., "A New Global Database of Lunar Impact Craters >1–2 km," Vol. 124, 2018, pp. 871–892. https://doi.org/https://doi.org/10.1029/2018JE005592.

[9] Pedregosa, F., Varoquaux, G., 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., and Duchesnay, E., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, Vol. 12, 2011, pp. 2825–2830.

[10] INRIA, "scikit-learn," , 2021. URL https://scikit-learn.org/stable/.

[11] Liu, J., "visualize-neural-network," , 2018. URL https://github.com/jzliu-100/visualize-neural-network.