Stellar Classification based on Numerous Characteristics using Machine Learning

Roberto Tamez

***Abstract*—The task of stellar classification can be tedious and lengthy when done manually. One can expedite stellar classification by creating an artificial intelligence model to automate the process. As we as a species continue to explore the frontier of the observable universe, we should seek to automate time intensive problems like stellar classification. The current stellar classification model serves to effectively categorize stars for research purposes regarding their distribution around the universe, so automating the development of this resource would allow professionals to allocate more time to explore the bounds of our current understanding of space and the universe. After finding and analyzing a dataset containing numerical and categorical features, a supervised learning approach was then used to train and test different models on their ability to classify the stars in the given test set. A Decision Tree Classifier, Random Forest Classifier, Ridge Classifier, and Support Vector Classifier were trained and tested using the data. The most successful models were the Decision Tree Classifier and Random Forest Classifier, each with about a 94 percent prediction accuracy across different accuracy metrics on the test data. Despite some drawbacks in regards to the availability of usable data, four models were trained and two were proven to be consistently and successfully accurate. Any future attempts at developing models for stellar classification should concentrate more on gathering data as to have a more thoroughly trained set of models.**

I. Introduction

In the field of astronomy, stellar classification, a categorical classification for stars based on their characteristics, needs a more consistent and efficient solution rather than continuing to categorize manually. Given qualities like luminosity, size, color, etc., the spectral class (a man-made systematic label given to a star based on its characteristics) of any star can be determined. Space exploration will likely continue until the end of human existence, so as we view further into space, problems that can be automated, like stellar classification, should be automated so that people in the field can focus on the newer, or more intensive, aspects of their research. As technology continues to advance, more and more stars are being discovered, and a new solution to classification needs to be developed. The stellar classification scheme is based on giving strict labels to values that lie on gradients of variability; given that many stars can lie between two spectral classes, a holistic evaluation of all of their attributes is necessary. This sort of evaluation cannot be done with basic, sequential code, and artificial intelligence is the best approach. Although the data of star characteristics is numerical and categorical, label encoding was employed to transform qualitative/natural language data into quantitative/numerical data for ease of use. Utilizing a supervised learning approach, said data can be used to classify a spectral class label for any given star.

II. Background

As described reference [1], the stellar classification scheme used today is a combination of the Harvard system and the 'MK' system of classification, each introduced in the 1800s and 1900s, respectively. The Harvard system primarily used surface temperature, and the MK system used luminosity, as features to classify the stellar class. Now, a mix of the systems is used to account for more characteristics when classifying a star.

An article in the Chinese Journal of Physics by Wen Xiao-Qing and Yang Jin-Meng in 2021 used machine learning to classify stars, galaxies, and quasi-stellar objects [2]. In their discussion of stellar classification, they found more success with a Random Forest Classifier and a Support Vector Classifier, and less success with a K Nearest Neighbors approach and a Decision Tree Classifier. I experimented with SVC, Decision Tree, and Random Forest, and found that my Random Forest model, similar to their results, performed very well. My Decision Tree, however, performed very well, which did not correspond with their results, and my SVC performed mediocrely, which contrasts their results also. Despite these discrepancies, their results helped give an idea as to how the results of this implementation might look when classifying. Also, it was stated that their accuracy scores were "always greater than 0.5". This seemed like a very low threshold that limited their models in sometimes being as useful as a coin flip, so I went in with this idea in mind. Despite being a very methodical and well-developed approach, their accuracy results were unsatisfactory, so there is a large area for improvement.

III. Data

To train my models, I used a kaggle dataset named Star Type Classification that contained 7 features of data for 240 stars [3]. All features of data were then graphed using a histogram or a countplot that shows the distribution of values per feature and will be shown after their respective explanations. The Random Forest Classifier, and the Decision Tree Classifier (the most successful models) were both trained using all of the features simultaneously, and then performed K-cross validation, or simply re-running each model with all features except for one at a time, to better analyze each individual feature's impact. Difference in prediction accuracy without those features taken into account by the models is listed during each of the feature's explanations.

A. Temperature

The first feature was temperature. Measured in Kelvin, this feature listed the numerical average surface temperature for a given star. The histogram shows that most temperatures lie below 5000 Kelvin with the rest of the values being more sparsely distributed from that to 40000 Kelvin. Without this feature of the data, the Random Forest Classifier performed significantly worse, and the Decision Tree performed marginally worse.

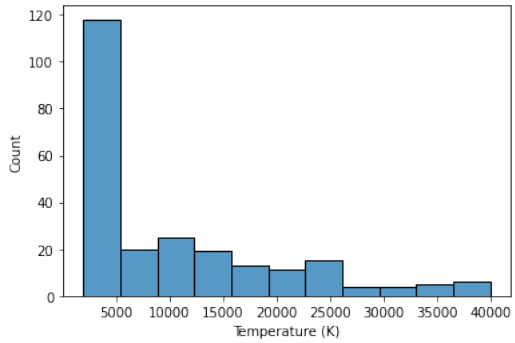

Fig. 1: Surface Temperature Counts

Source: Primary

B. Luminosity

Second was luminosity. Measured in luminosity per solar luminosity, a numerical luminosity value is listed for every star. These values ranged widely from the ten-thousandths to the hundred thousands, but the grand majority are under 50000 L/Lo. Without this feature of the data, the Random Forest Classifier performed slightly worse, and the Decision Tree's performance became very inconsistent (performance ranging from much worse to slightly better than normal).

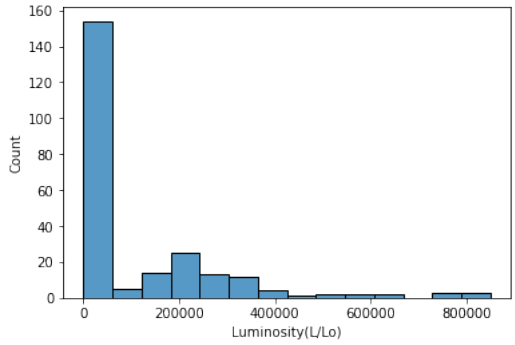

Fig. 2: Luminosity Value Counts

Source: Primary

C. Radius

Third is radius. Each star was given a numerical value measured in radius per solar radius. Almost all values lied under 100 R/Ro with the remaining few ranging to 2000 R/Ro. Without this feature of the data, the Random Forest Classifier performed the same, and the Decision Tree performed the same, which leads one to believe that the radius may not have a significant impact on spectral class determination.

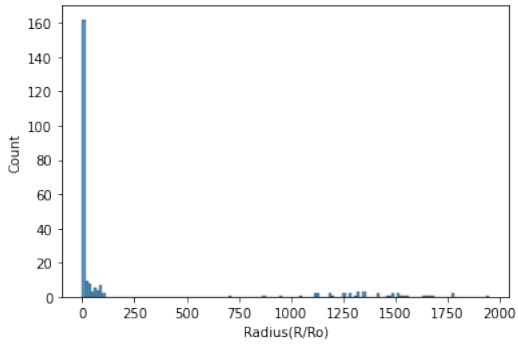

Fig. 3: Radius Value Counts

Source: Primary

D. Absolute Magnitude

Fourth is absolute magnitude, measured in absolute visual magnitude units (MV). This numerical unit is based on the standardization of the apparent magnitude of a star as viewed from a distance of 10 parsecs (a parsec being a unit of distance approximately equal to 3.26 light years). Apparent Magnitude is simply a classification system to measure the brightness of a star. The distribution of the stars' magnitudes is bimodally clustered with many lying around the -7 and 15 areas. Without this feature of the data, the Random Forest Classifier performed the same, and the Decision Tree performed slightly worse.

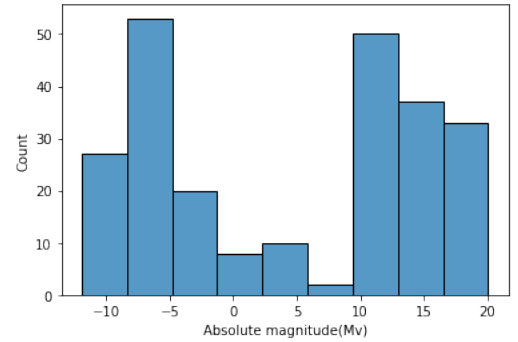

Fig. 4: Absolute Magnitude Counts

Source: Primary

E. Star Type

Fifth is star type. In this dataset there are 6 categorical star types: 0 - Brown Dwarf, 1 - Red Dwarf, 2 - White Dwarf, 3 - Main Sequence, 4 - Supergiant, and 5 - Hypergiant. Each text feature was converted into the previous integers using a label encoder. This distribution is perfectly uniform. Without this feature of the data, the Random Forest Classifier performed the same, and the Decision Tree performed the same. This is interesting as the star type would intuitively seem to be a very significant characteristic of a star, but the results suggest that the star type is not that significant in determining stellar class.

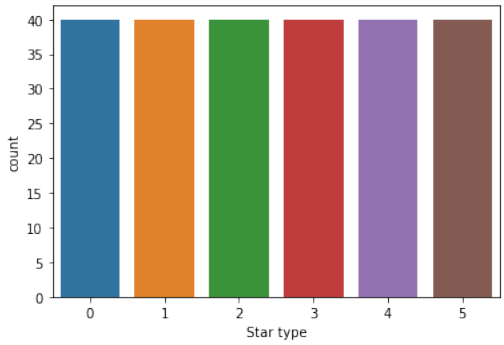

Fig. 5: Star Type Counts

Source: Primary

F. Star Color

Sixth is categorical star color. Although there are 19 star colors in this data set, most are either Red or Blue. Without this feature of the data, the Random Forest Classifier performed significantly worse, and the Decision Tree performed marginally worse, showcasing that color is a very significant factor in determining stellar class.

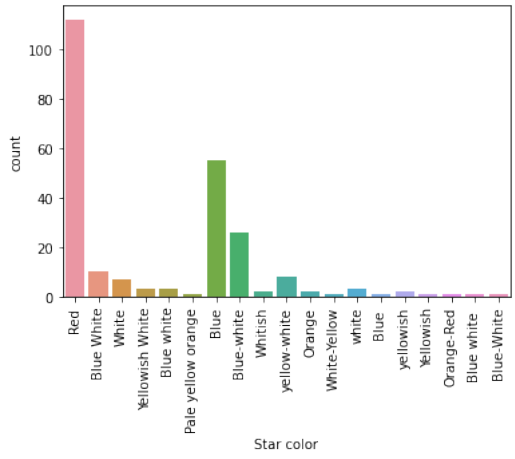

Fig. 6: Star Color Counts

Source: Primary

G. Spectral Class

Lastly is the spectral class. This is the categorical feature that my model aims to predict based on the other 6 features. A star's spectral class can be either O, B, A, F, G, K, or M. The majority of stars are M, then B and O are closely tied for the second most abundant, and the rest are less than common. After encoding, the spectral classes were relabeled as the following: A - 0, B - 1, F - 2, K - 3, M - 4, O - 5.

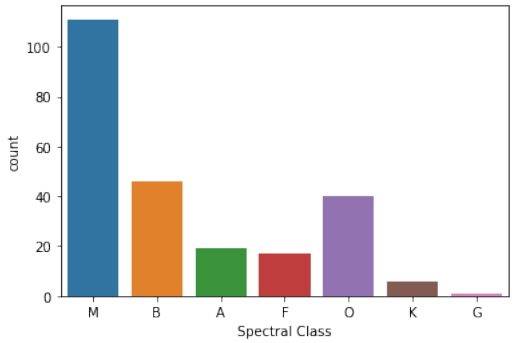

Fig. 7: Spectral Class Counts

Source: Primary

Due to only having one G class star, this star was omitted and removed as there was no way to have a G class star in both the train and test sections of the dataset. All of the categorical data was encoded to become numerical and then the dataset was split into 80% train and 20% test sets. All features in the dataset were utilized as stellar classification is a holistic classification. Star radius and type oddly seemed to not have too much of a significant impact. A possible explanation for the radii's insignificance could be due to the dataset having an exceptionally skewed set of radii. Aside from this exception, all of the given characteristics are necessary to consistently and accurately predict a star's spectral class.

IV. Methodology

To begin an approach for creating appropriate models for stellar classification, I began by first visualizing the data by using graphs as can be seen in the dataset section of this paper. Upon splitting the dataset into train and test sets, problems arose due to the lone G class star. Due to only having one star with the G spectral class, the star was removed from the dataset as there would be no way to test for the accuracy of predicting a G class star. If the star went into the training data, then there would be no way to test for the model's accuracy when classifying a G class star; And if the star went into the testing data, then the model would have to predict the class of the star with absolutely no training in regards to the G stellar class. The data was then successfully split into 80% train and 20% test sets as this split provided the most consistently accurate results. The training data contained 191 stars and the test data contained 48 stars. To have only quantitative data, a label encoder was fit and used to transform the star color and spectral class features of the data set as they

were the only features that used natural language. Then, with all features being strictly numerical, the training of the models began. The models trained using this dataset were a Random Forest Classifier, a Decision Tree Classifier, a Ridge Classifier, and a Support Vector Classifier. An accuracy score, precision score, recall score, and f1 score were calculated (precision, recall, and f1 were run with the average set to 'weighted' as a parameter) for each model and a confusion matrix was plotted to visualize the results (listed and shown in the results and discussion).

A. Classification using a Decision Tree Classifier

One of the most successful models was the sklearn Decision Tree Classifier [4]. According to reference [5], The Decision Tree classifier works by forming a tree of logic gates called 'nodes' where the parameters are determined by the model itself. These logic gates help to slowly separate data points and will typically stop on a depth given through the parameters. If the tree depth is allowed to reach 100 percent purity, in other words it completely separates both types of training data points, then the model could potentially have problems with overfitting. Below is a visual representation of a Decision Tree Classifier with a more relatable set of conditions for every node and leaf (real Decision Tree splits will not always be as interpretable to humans).

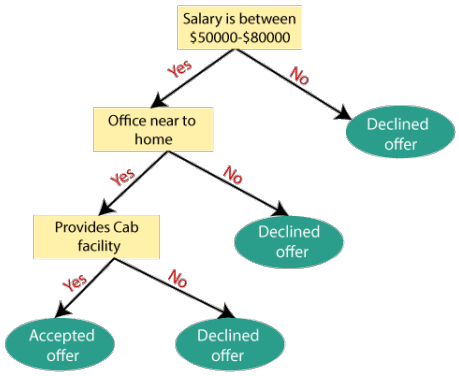

Fig. 8: Basic Decision Tree

Source: [6]

The Decision Tree Classifier default parameters of an unbounded max_depth and 2 min_samples_split were sufficient and any hyperparameter tuning led to a decrease or no change in prediction accuracy. The Decision Tree Classifier was slightly inconsistent as it typically predicted with a slightly higher accuracy when compared to the Random Forest Classifier, but other times it performed slightly worse. The cause for this variability was because the Decision Tree Classifier contains a random_state (for the algorithm that selects max_features - a parameter used for the nodes) that was not set.

B. Classification using a Random Forest Classifier

Another, more consistent, successful model used was the sklearn Random Forest Classifier [7]. The Random Forest Classifier works by generating multiple Decision Trees (previously explained in the 'Classification using a Decision Tree Classifier' subsection) and then using the predictions of those Decision Trees to 'vote' for what the most likely, final, prediction is [8]. Below is a basic visual representation of a Random Forest Classifier.

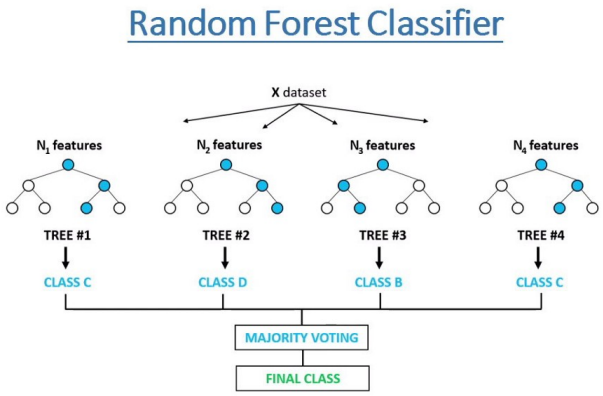

Fig. 9: Random Forest Classifier Diagram

Source: [9]

The default hyperparameter of 100 n_estimators proved to be very accurate and any fine tuning resulted in either a loss or no change on the accuracy of the model.

C. Classification using a Ridge Classifier

The third model used was the sklearn Ridge Classifier [10]. Defined by reference [11], a Ridge Classifier is a type of linear regression that works by converting the soon to be predicted values into a range of (-1, 1). Then, rather than just fitting a least squares regression line, the classifier adjusts the slope and position of the line of best fit using a set of parameters to more accurately extrapolate for the test data. The hyperparameters would then be tuned to more properly set this displacement for better accuracy. Below is a picture that visually represents the displacement of the least square regression line to form the ridge regression line of best fit.

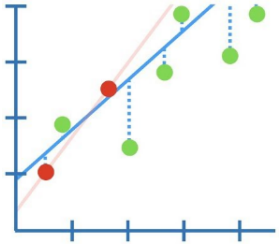

Fig. 10: Ridge Classifier Visual

Source: [11]

Unfortunately, the Ridge Classifier yielded unfavorable results and no hyperparameter tuning had any significant effect on the accuracy of the model, so the model was run using the default hyperparameter of alpha = 1.0. Most accuracy scores were below 80%, which is subpar when compared to the other models tested.

D. Classification using a Support Vector Classifier

Lastly, a sklearn Support Vector Classifier model was used [12]. Defined by reference [13], a Support Vector Classifier, or SVC, is a relatively simple model that attempts to separate data points based on the given features with a 2 dimensional line or a 3 dimensional plane in a coordinate plane. Based on this predicted 'boundary', the labels for the input testing points are predicted depending on what side of the line, or plane, that they lie on. Below is a visual representation of what a basic Support Vector Classifier would look like.

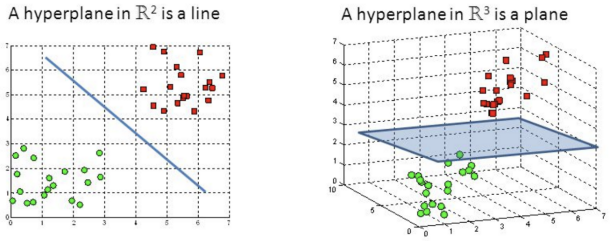

Fig. 11: Support Vector Classifier Visual

Source: [14]

With the default parameter of an rbf kernel, the SVC initially performed very poorly, but upon setting the kernel hyperparameter to "linear", the model began to perform significantly better. Although the SVC showed promise, the Decision Tree and Random Forest outperformed it by a relatively notable amount.

V. Results

A. Random Forest Classifier Results

The Random Forest Classifier was one of the most consistently better models with a 0.9375 accuracy score, 0.9188 precision score, 0.9375 recall score, and a 0.9259 f1 score. All but the accuracy score had "average = 'weighted'" passed as a parameter for evaluation. The default hyperparameters from the sklearn Random Forest Classifier performed the best, and any hyperparameter tuning led to either a decrease or no change in accuracy. As can be seen below, a confusion matrix was plotted based on the model's predictions on the test set.

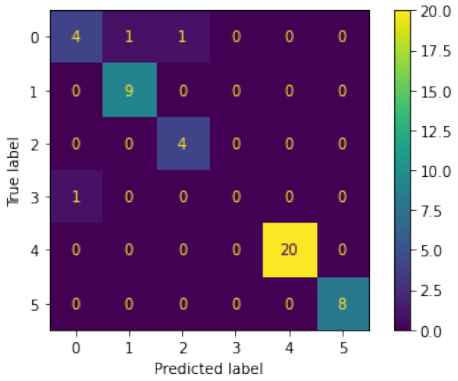

Fig. 12: Random Forest Confusion Matrix

Source: Primary

The confusion matrix depicts that the Random Forest Classifier generally did very well, but had some trouble predicting the 0 and 3 classes, or the A and K stellar classes respectively. It can be assumed that the model performed poorly when predicting the K class as there were not many K class stars to train off of in the training data, but it is unclear as to why the model had such trouble when predicting the A stellar class stars.

B. Decision Tree Classifier Results

The Decision Tree Classifier typically performed with an equal, if not higher, degree of accuracy depending on the score used. At times however, the Decision Tree would dip slightly below the Random Forest Classifier in accuracy, so this model is not as consistent. The Decision Tree Classifier scored a 0.9375 accuracy score, 0.9531 precision score, 0.9375 recall score, and a 0.9341 f1 score. All but the accuracy score had "average = 'weighted'" passed as a parameter for evaluation. The default parameters worked best for the Decision Tree as any hyperparameter tuning led to a decrease or no change in prediction accuracy. Below is a confusion matrix that was plotted based on the model's predictions on the test set.

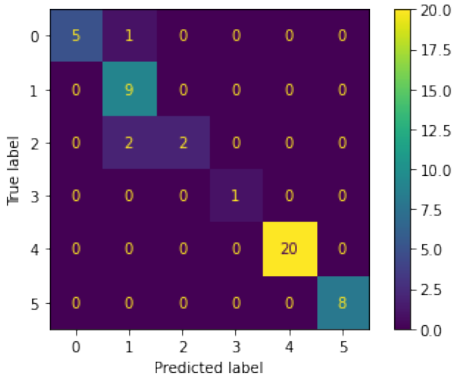

Fig. 13: Decision Tree Confusion Matrix

Source: Primary

The confusion matrix accurately depicts the Decision Tree's competency when classifying. It only had issues when classifying the 0 and 2, or the A and F. classes, but still only came within one stellar class off in each incorrect prediction. The model seems to have a preference for the 1, or B, class as the only predictions it missed went to that category. Besides this inaccuracy, the Decision Tree appears to be the most capable model as it not only had a high prediction accuracy, but out of the predictions it missed, it only missed them by a single stellar class. This slight imprecision suggests that the model is only inaccurate by minute levels of variability, as opposed to the larger inaccuracies in the rest of the models.

C. Ridge Classifier Results

The Ridge Classifier was the worst performing model with a 0.7292 accuracy score, 0.8699 precision score, 0.7292 recall score, and a 0.7858 f1 score. All but the accuracy score had "average = 'weighted'" passed as a parameter for evaluation. Majority of hyperparameter tuning led to a decrease or no change in accuracy, but the only hyperparameter tuning that increased accuracy was setting the class_weight to 'balanced', which is what the model was trained using. Below is a confusion matrix to illustrate the Ridge Classifiers predictions on the test set.

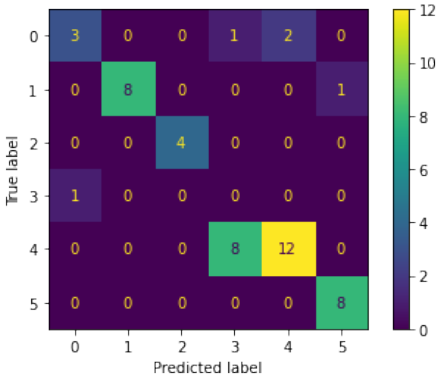

Fig. 14: Ridge Classifier Confusion Matrix

Source: Primary

As can be easily observed, the Ridge Classifier performed very poorly when predicting a grand portion of the stellar classes. The model was only completely successful in predicting the 2, or F, stellar class, but severely mispredicted many stars among the rest of the categories. There could be many reasons as to why the model predicted most categories so poorly, so it is hard to speculate as to why.

D. Support Vector Classifier Results

Lastly, the Support Vector Classifier performed moderately well with a 0.8959 accuracy score, 0.8699 precision score, 0.7292 recall score, and a 0.7858 f1 score. All but the accuracy score had "average = 'weighted'" passed as a parameter for evaluation. Upon tuning the hyperparameters, changing the kernel to 'linear' (from the default value of 'rbf') had a tremendously positive effect on prediction accuracy. Besides that, no other hyperparameter tuning positively affected the model. The confusion matrix below was plotted based on the SVC's predictions on the test set.

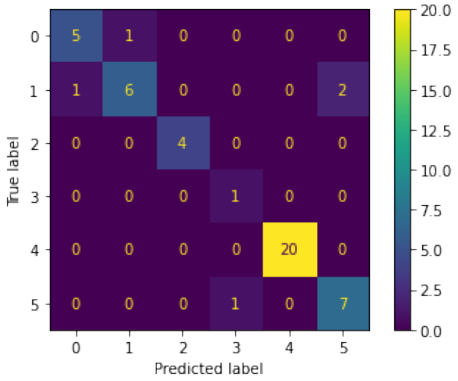

Fig. 15: Support Vector Classifier Confusion Matrix

Source: Primary

As can be seen, the SVC was relatively consistent with its predictions, but severely misclassified some of the 1, or B, class stars and incorrectly predicted a few others. The misclassification of the B class stars was the majority of the inaccuracy in the model's predictions, potentially due to the model not having been well trained with the B class stars, or maybe due to the stars lying near the classification boundary for the class. Overall, the SVC performed decently, but the Decision Tree and Random Forest classified with a notably higher degree of accuracy.

VI. Conclusion

As space exploration continues to develop, more time should be invested in automating simpler processes so as to not draw time and attention away from the more manual and time intensive work that professionals are occupied with. My attempt at said automation resulted in having two very accurate models for star classification, a Decision Tree Classifier and a Random Forest Classifier. The structure of the leaf-node hierarchy proved to efficiently and accurately determine the stellar class of the given stars, for they determine stellar class as the system was meant to work: by analyzing a single feature individually at a time. In other words, the structures of the Decision Tree and Random Forest classifiers emulate the sequential process of manual classification. The primary hindrance to achieving higher accuracy scores on the models was the lack of data available. Only having 240 stars to train and test the models proved disadvantageous, so any successive attempts at the problem of stellar classification should be done with larger datasets of star characteristics to better train and more effectively test the models. Also, the lack of G class stars to train and test was a significant barrier to the development of the model and data including more G class stars should be employed.

ACKNOWLEDGMENTS

I would like to acknowledge Sophia Barton for making this paper possible and for the expertise she provided as I wrote this paper. I would also like to thank Inspirit AI for providing this opportunity to learn about the process of making a research paper.

REFERENCES

[1] "Stellar Classification." Encyclopædia Britannica, Encyclopædia Britannica, Inc., www.britannica.com/science/stellar-classification.

[2] Xiao-Qing, Wen, and Yang Jin-Meng. "Classification of Star/Galaxy/QSO and Star Spectral Types from LAMOST Data Release 5 with Machine Learning Approaches." Chinese Journal of Physics, vol. 69, 2021, pp. 303–311., doi.org/10.1016/j.cjph.2020.03.008.

[3] Paulycyclic>. "Paul Bacher: Notebooks Expert." Kaggle, www.kaggle.com/paulbacher/code.

[4] "Sklearn.tree.decisiontreeclassifier." Scikit, scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html.

[5] "Decision Tree." CORP-MIDS1 (MDS), 11 July 2022, www.mastersindatascience.org/learning/machine-learning-algorithms/decision-tree/#:∼:text=A%20decision%20tree%20is%20a,that%20contains%20the%20desired%20categorization.

[6] "Decision Tree Algorithm in Machine Learning - Javatpoint." www.javatpoint.com, www.javatpoint.com/machine-learning-decision-tree-classification-algorithm.

[7] "Sklearn.ensemble.randomforestclassifier." Scikit, scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

[8] Meltzer, Rachel, et al. "What Is Random Forest? [Beginner's Guide + Examples]." CareerFoundry, 15 July 2021, careerfoundry.com/en/blog/data-analytics/what-is-random-forest/#:∼:text=2.,group%20than%20they%20do%20alone.

[9] Chauhan, Ankit. "Random Forest Classifier and Its Hyperparameters." Medium, Analytics Vidhya, 23 Feb. 2021, medium.com/analytics-vidhya/random-forest-classifier-and-its-hyperparameters-8467bec755f6.

[10] "Sklearn.linear_model.Ridgeclassifier." Scikit, scikit-learn.org/stable/modules/generated/sklearn.linear_model.RidgeClassifier.html.

[11] Starmer, John. "Regularization Part 1: Ridge (L2) Regression." YouTube, YouTube, 24 Sept. 2018, www.youtube.com/watch?v=Q81RR3yKn30.

[12] "Sklearn.svm.SVC." Scikit, scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html.

[13] "How SVM Works." How SVM Works, www.ibm.com/docs/en/spss-modeler/saas?topic=models-how-svm-works.

[14] Gandhi, Rohith. "Support Vector Machine - Introduction to Machine Learning Algorithms." Medium, Towards Data Science, 5 July 2018, towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47.