University of North Carolina-Charlotte

Stellar Classification

Primary Paper: Stellar Classification by Machine Learning [1]
Author: Zhuliang Qi
Year: 2022

Kevin Avila

Applied Machine Learning (ITCS 5156)

Dr. Minwoo Lee

October 12, 2022

1 Introduction

Stellar classification is something that is vital to the advancement of humans for both our drive to learn, and our survival. Without knowing what stars did, or how they acted, we would never know that our sun is a ticking time bomb. Knowing the patterns and life stage that each star is in can help us in understanding why things are the way they are as well giving us the correct precautionary measures for if a star dying were to ever become a serious threat.

1.1 Motivation and Challenges

Space has always been something that I have been very interested in, but have not had the chance to work with things related to it given my major. Now with this project, it gives me a great chance to learn more about space, as well as helping me improve my machine learning skills that I have learned throughout the semester. The challenges with working with these kinds of datasets can be the computation power that is required for them. Unfortunately, the dataset that I was originally going to use, I was no longer to access so I had to adapt and find a new dataset. This new dataset did not have much data to work with, hence it is something that I wish to improve on in the future.

1.2 Summary of Approach

The approach that was taken was using three different models. These models being a Decision Tree, Random Forest, and Support Vector. The main reason behind why these models were picked was because of how I did not really get a great understanding of the Decision Tree and Random Forest model through the class. This gave me an opportunity to get a better understanding of how and why they work. Lastly, the reason that SVM was chosen was because this was my favorite model that I learned in this class and wanted to see how it would perform in a One Vs All approach.

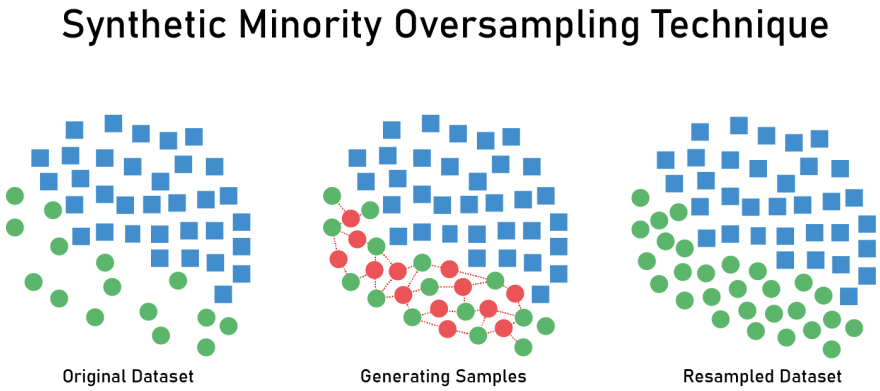
2 Backgrounds/Related Work

2.1 Stellar Classification by Machine Learning [1]

Zhuliang Qi is the author of this paper and used a very interesting dataset from the Sloan Digital Sky Survery (SDSS.) This dataset dealt with classifying things related to astronomy (galaxies, quasars, and stars) as opposed to classifying the type of star something is, and is also very large. In fact, there are 59,445 examples of galaxies, 21,594 examples of stars, and 18,691 examples of quasars.

In order to level out the playing field so that the training model can effectively learn the decision boundary, a SMOTE (Synthetic Minority Oversampling Technique) function was used. This

SMOTE function essentially just creates new data points that are not duplicates in order for the classes to be more evenly distributed.



Once the preprocessing was done, the models that were used were the Decision Tree, Random Forest, and SVM. For some quick numbers, all models had above a 95% precision, recall, and F1 score, but Random Forest performed the best of the three models. This Random Forest even had a 100% training accuracy when classifying stars. When talking about testing accuracy, Random Forest had the best with a 98% accuracy followed by both the Decision Tree and SVM with 97% accuracy.

Some pros of this paper is the dataset that they use. Firstly, the SDSS is an extremely reliable source of information and is one of the largest astronomical organizations that exists. Secondly, the amount of information that is in this dataset really gives you an idea of how well the models can be trained given the size of it. However, this can also be a con depending on how you implement the data. It was vital for Qi to add the SMOTE function to the preprocessing or else the models would have a bias towards galaxies since there was about 40,000 more of those than stars or quasars.

This data relates to my approach since I used the same models that Qi used in order to classify my dataset. It was great to read about how these models work (thoery) and how they scored with a dataset this large.

2.2 Classification of star/galaxy/QSO and star spectral types from LAMOST data release 5 with machine learning approaches [2]

This paper uses data from LAMOST which is a telescope named after a 13th century Chinese astronomer. This dataset is focused on classifying stars, rather than classifying galaxies, quasars and stars. What is different about this paper though is that they use the K Nearest Neighbor model on top of the Decision Tree, Random Forest and SVM.
For data preprocessing, the authors must first get rid of the noisy data. This is done by setting a threshold for magnitudes of J or K equal to or lower than 10. J and K in this example is

photometry from the Micron All Sky Survey. Something similar that these authors did is using a SMOTE function to even out the data as they had a lot more stars of class G compared to the rest of the classes. According to the reported accuracies from when they tested the models on a dataset with galaxies, quasars and stars, KNN had the highest accuracy with 97% with SVM, Random Forest, and Decision Tree following in that order. On the other hand, when classifying only stars, Random Forest had the highest accuracy with 72.5% accuracy followed by SVM, KNN, and Decision Tree in that order.

Some pros of this paper are similar to the ones from the previous paper. They use a very reliable database and use large amounts of data to be trained. A con that I see right off the bat though is that they are attempting to use the same models with the same parameters for both sets of data. This includes classifying galaxies, quasars, and stars and classifying only star types. Because these are two different things, new hyperparamaters should be used since features for these datasets are completely different. Another thing that I saw is how they handled the noisy data. I think that maybe they removed some important information that would help the models in classifying the star types, but maybe not in galaxies, quasars, and stars which is why it was removen. These things should have been completely separated in order to ensure that the model can perform its best.

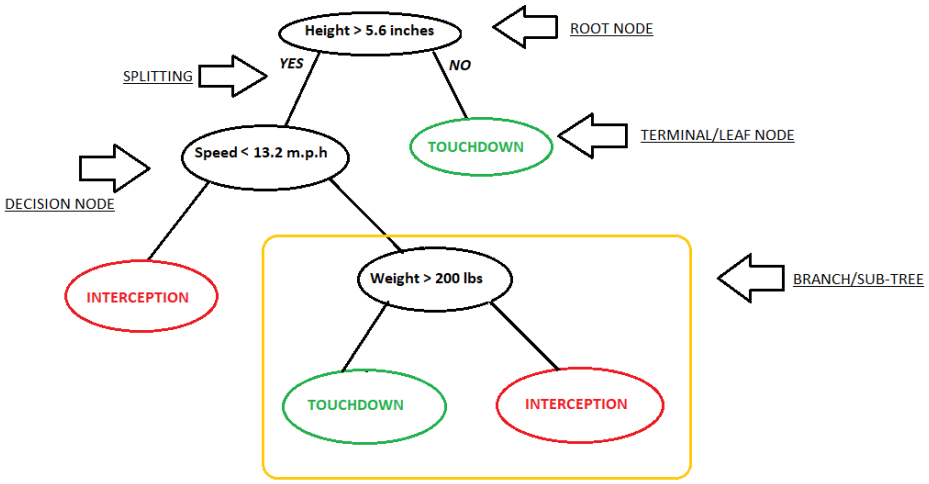
This data relates to my approach since it uses most of the of the models that I decided to implement. Those models being the Decision Tree, Random Forest, and SVM.

3 Methods Used

The methods that were used were the Decision Tree, Random Forest, and SVM models based on Qi's paper. The reason that these models were chosen were explained earlier, but as a quick summary it is because I wanted more practice/understanding with Decision Tree and Random Forest. The reason that the SVM was chosen was because that was my favorite model learned in this class.

3.1 Decision Tree

A Decision Tree is a supervised machine learning model that is mostly used for classification and can be used for multi-class classification. This algorithm splits the features of a dataset through something called a cost function. There is a root node, nodes in between the root and leaf, and as just mentioned, leaf nodes. Leaf nodes are what correspond to a prediction and the path from the root to the leaf node are the tests that were passed in order to get to that leaf.



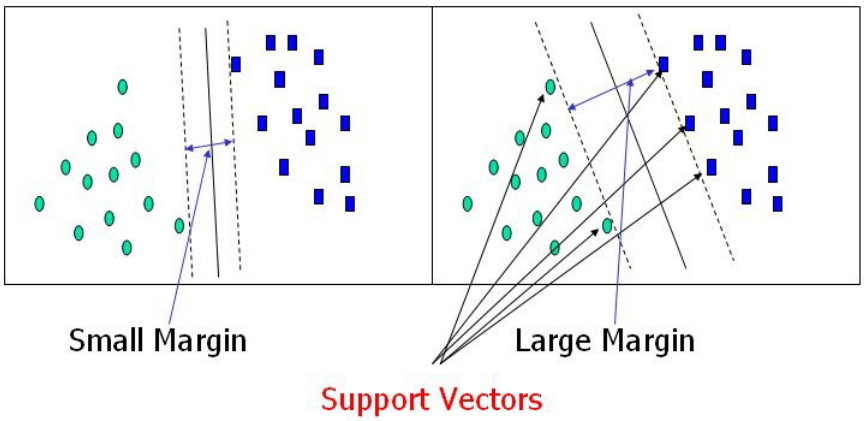
By looking at this figure, the process of how a classification is taken place. First, it check if the height is greater than 5.6 inches. If not, then it is a touchdown, and if not this process will continue until a classification was reached.

3.2 Random Forest

The Random Forest model is also a supervised machine learning model that once again is mostly used for classification, but can be used for regression as well. Now that the Decision Tree is understood, it becomes a lot easier to understand what a Random Forest is. A Random Forest is essentially multiple Decision Tree models put into one. Some benefits to using the Random Forest model over the Decision Tree model is that it reduces overfitting while also increasing the precision. The outcome in the Random Forest model comes from the mean output of trees within the model. Something important to note is that the more trees there are, the more accurate your model will be. This comes with the drawback of computation speed being decreased significantly if the number of trees is very high.

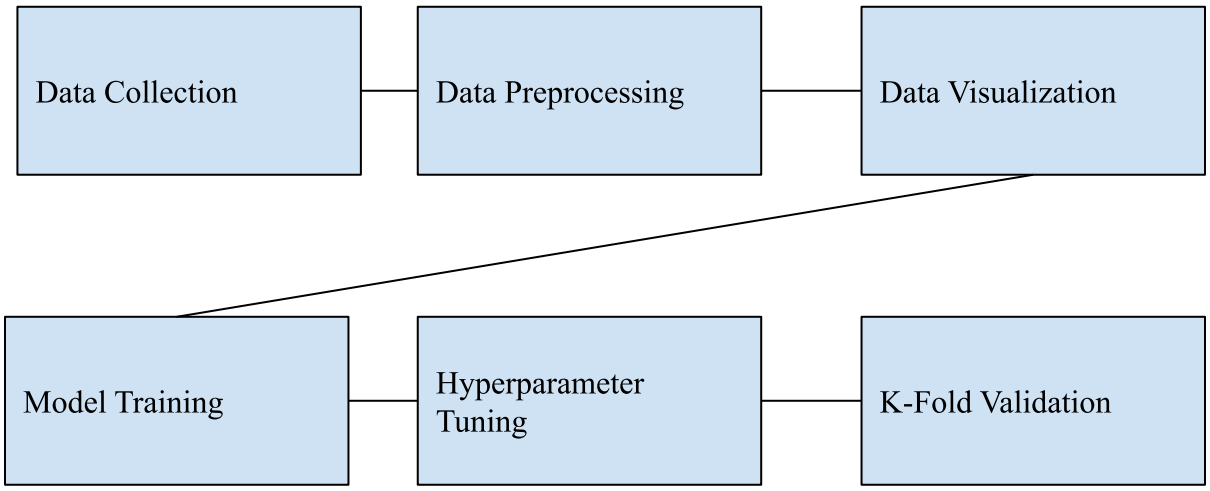
3.3 Support Vector Machine

The Support Vector Machine model is another supervised machine learning model that is used mainly for classification. An important thing to note about this model is that it is not inherently multiclass, so the approach that I used in order for it to work with my dataset is the One vs All approach.



For linear seperable datasets, there are an enormous amount of hyperplanes that can be found, but only one has the largest margin. This is the objective of SVM; maximize the distance or margin of the classifier in the feature space so that it performs the best.

3.5 Overall Architecture



4 Experiments

My experimental setup consisted of data collection, data preprocessing, data visualizaiton, training, hyperparameter tuning and validation. These are basic steps in any machine learning architecture, and thought that it would be a good place to start for my first project in machine learning.

4.1 Test Results

My test results were somewhat similar to the paper by Qi, with the most similar thing is how the Decision Tree and Random Forest performed very well with 97.9% accuracy on the testing data. What was extremely different was my result with the SVM model having only 41.6% testing accuracy. This is almost a 60% accuracy difference from Qi's SVM model. Something worth noting is that this is before hyperparemeter tuning, which is not said if it was done in the paper by Qi. After hyperparemeter tuning, I was able to significantly improve the accuracy to 97.9% accuracy, exactly the same as the Decision Tree and Random Forest model. A One vs All approach was also used for the SVM since it is natively binary.

K-Fold Validation was also used on the data in order to test out how well it worked, and to help with avoiding overfitting. In my code, I used 3-Fold Validation and achieved better results for my Decision Tree and Random Forest model with 99.48% accuracy. I also used 3-Fold Validation for my SVM model, but with this it only achieved 82.29% accuracy which I was somewhat disappointed about.

4.2 Analysis of Results

As stated earlier, the results were similar to the results achieved by Qi after some tuning for the SVM model. Another set of results that are important to mention is the fact that the Decision Tree and Random Forest achieved 100% accuracy when using training data. At first, I was worried about the model actually being overfitted, but once I saw that it still performed extremely well on the testing data, I concluded that was not an issue. I believe the reason that the accuracy is so high with this data is because the problem itself is not too difficult for the models to figure out. The Decision Tree is able to successfully break apart the features and the SVM is able to find a great hyperplane where the margin is maximized. One thing that is a big coincidence though, is how the accuracy of all three models are exactly the same.

By looking at the confusion matrix, we can also see that all three models predicted that a particular star was of type 1, when it was actually of type 3. Something that I attempted to figure out was if all three models were miscalssifying the exact same star. Unfortunately, I was not able to do that, but if I was able to figure it out and it was the same star, I do have an explanation for that. If the models were to misclassify the same star, it would be because the star is close to transitioning the another star. This would mean that the features of the misclassified star is "overpowering" the features of the true label star. In other words, certain features hold more weight than others, hence resulting in all three models misclassifying the same star.

The reason I chose 3-Fold Validation was just because of the computation power that I had at the moment. Ideally, I would choose more, but even with 3 folds, I think the accuracies still somewhat improved. Something that really interested me was the SVM accuracy going down after cross validation, which is not something I expected. I believe the reason that K-Fold Validation made the accuracy go down is because of overfitting, but it seems a bit odd that it would still perform well on the testing accuracy. I think in order to counter this, I would need to get more training data as well as doing some feature engineering

5 Conclusion

I believe that I did a good job with my implementation of this dataset, but I definitely have improved. I think something that really lacked with my implementation is the dataset that I used. Originally, I wanted to use a smaller dataset from the SDSS, but did not download it and lost access to it. With this dataset, I would have been able to do a lot more data preprocessing, and just be able to train my model with a lot more data. As they say when talking about machine learning, the more data the better. In terms of what I would improve with my implementation of the dataset is maybe using a SMOTE function to have more data just like Qi did to even out his classes.

I think this project helped me understand Decision Trees and Random Forests more since I was very confused on them during the week we learned about them. I also learned just how much hyperparameter tuning can help a model as seen by the almost 60% jump from the SVM model with no tuning vs the SVM with tuning. It also helped me with learning about how to adapt when things happen such as what happened with the dataset that I wanted to use originally. This is a great thing to learn as it can be applied to so many aspects of life, not just machine learning.

6 Contributions

My contributions to this code was the data visualization as well as both SVM models (one with hyperparameter tuning and one without.) I also added the classification report and confusion matrix to not only just see that accuracy scores on training and testing sets. The source code [3] is obtained from a Kaggle user and the dataset [4] is obtained from Kaggle as well.

7 References

[1] Qi, Zhuliang. “Stellar Classification by Machine Learning.” *Research Gate*, 2022, https://www.researchgate.net/publication/362955934_Stellar_Classification_by_Machine_Learning

[2] Wen, Xiao-Qing, and Jin-Meng Yang. “Classification of Star/Galaxy/QSO and Star Spectral Types from LAMOST Data Release 5 with Machine Learning Approaches.” *Science Direct*, 2021, https://www.sciencedirect.com/science/article/abs/pii/S0577907320300605

[3] Shah, Yash. “Star-Type classifier” *Kaggle,* 2022, https://www.kaggle.com/code/yash161101/star-type-classifier/notebook

[4] Baidya, Deepraj. “Star dataset to predict star types” *Kaggle,* 2022, https://www.kaggle.com/datasets/deepu1109/star-dataset