Wine Not? – A Classification Data Translation Challenge

By Rosalie Sherry
OMSBA5067 Spring Quarter 2021

Introduction

As someone who has grown up around wine my whole life, I was instantly drawn to the wine dataset from the UCI Machine Learning Database. I enjoy wine and tend to enjoy red wines that either are Cabernet Sauvignon or have a large variety of those grapes in it. The type of grape that makes up a wine, is the cultivar, the goal in which this dataset hopes to classify through different machine learning techniques. My focus of this is when a wine is labeled a “red blend”, “table wine”, or “house wine” it is not always a given whether or not you know what grapes are present. My goal with this paper was to hopefully build a classifier that could help identify perhaps purposefully vague wine names so I would be more inclined to know if I like it or not.

Additionally, as I progressed working on this project I began playing around with sizes for the test and training dataset. This data translation challenge ultimately transformed from an attempt to find the “best” classifier to classify cultivars based on 13 features to an attempt to prove that the larger a train dataset is, the more accurate the test case is.

Literature Review

Upon choosing my dataset the second week of class I read up on different papers that implemented machine learning classifiers to gain inspiration for my own project. Originally my takeaway from these 5 papers were what classifiers people used and why, as I had not chosen what classifiers I wanted to test in my own dataset as well as the way they measured accuracy. Before learning about it in class these papers brought to my attention F-Measure and ROC as a way to measure a classifiers success as well as brought a heightened importance to me of accuracy, precision, and recall. *Should I have gone in the original direction of my proposed project, I would have used these* but due to pivoting during the testing phase these measures went out the window. Throughout the five papers the main classifiers they used was Naïve Bayes, Decision Tree, and Support Vectors so, following in these authors footsteps I knew that I would need to add these classifiers to my own project if I was going to get a full picture of what was “the best” classifier. After reading these papers, it was time to put what I had learned in action.

Methodology

As touched on in the prior two sections, my original goal for this Data Translation Challenge was to find the “best” classifier for the dataset I had chosen. I knew due to the literature review three of the classifiers I wanted to use but ended up settling on five. The two additional classifiers chosen were based off their differences from the other three. I wanted to get a good picture of which one was the “best” and seeing as they were rather simple it seemed right to add a polynomial regression in as the others were rather one dimension. Lasso was chosen despite being another linear model because it also regulates the data on its own. With these five classifiers, I felt as if I was approaching this problem from every possible direction in order to correctly conclude which classifier was “the best”. From there all I needed to do was choose a test and train data size and then figure calculate the various measures of their success in classifying the data.

I. Implementation

As I had chosen my five classifiers it was now time to further choose what type of each classifier I was using. For Naïve Bayes and Support Vector there was no additional thought needed in choosing them but the other three had other characteristics that needed to be chosen. In the file titled DTC.py is how I choose which characteristics to give each model. Splitting the data 50/50 for test and train I ran through every possible classifier characteristic (and characteristic combination) for the five chosen and whatever resorted in the highest score would be what was going to be used going further in. the project. The scoring measure used here was the score measure that is a part of the sklearn package seeing as all these classifiers came from that same package it seemed like the easiest way to make an early decision in the project's process. In the DTC presentation on slide 4, one can see the console view after running this initial test. This determined that for Decision Tree it would be ran with gini and a max_depth of , the polynomial regression would be run with a polynomial features of 2, and lasso would be run with an alpha of 2.

Next, was running through the classifiers and choosing the correct way to split the data between testing and training. Throughout this course we have discussed how it was necessary for training datasets to be larger than the testing dataset and for much of this class we have done an 80/20 split so I began there. However, I quickly discovered a large disparity between effectiveness of classifiers and wanted to dive in deeper. This is where the pivot in question went from, I wanted to assure myself that if I made the training dataset larger then it would be better at predicting the test dataset. Knowing that I didn't have a binary classifier this also sent out the initial idea of doing the scores out the window so again I settled on sklearn's built in score feature.

I had used a 50/50 split in order to determine what characteristics of classifiers to use and not wanting to double dip, I ruled out using that one in testing my working hypothesis. I then settled upon 2/3, 3/4, 9/10, and 19/20 as the size for the test dataset as they gradually got larger. Next, was to test.

II. Experimentation

It would not be a very good experiment to just run each classifier once so it was determined that for each classifier for EACH training/testing size combination the machine learning algorithm would run it 1,000 times then take the average score for each to compare to others in its respective group size. The choice for 1,000 was that running it that many times would hopefully cancel out a lot of noise because as I reflected on above when I was discussing the pivot in my DTC's business statement is that most of the classifiers are fairly bad at classifying the data. Classifiers like polynomial regression also had a high level of volatility in its scores throughout the project so the large number of tests aimed to get that managed.

III. Results

To answer the initial question that I aimed to answer, it appears that Lasso and Support Vectors were the "best" classifiers for determining cultivars in the wine dataset. Throughout the different combinations of test/train size it is in dispute of which classifier was better or not which can be seen in the folder Final Graphs as each graph for each size group is there. It is important to note that lower score is better. Naïve Bayes was by far the worst classifier for cultivars throughout the whole experiment, never getting an accurate prediction.

However, for the task I set out to prove using this data, that a larger training dataset leads to a higher level of accuracy, that appears to not have been proven nor disproven during this process. Though overall the errors tend to be getting smaller in the test dataset as the train gets larger, they are not linearly decreasing. For example, if one looks at the TestError0.05 and compares it to TestError0.10 they'll see that all the scores for 0.05 are slightly larger than for 0.10. However, if one compares 0.10 to 0.33 they'll see that Lasso and Support Vector scored better between the two while the other three actually scored better at 0.33.

Conclusion

This data translation challenge made me want to dive deeper into the actual process of training and testing data. Looking at the results and seeing the scores vary so much between the different test/train combinations makes me curious why. I wonder in part if Naïve Bayes, Decision Tree, and Polynomial Regression scored better at a lower training level might have something to do with overfitting for those three models rather than them just being bad classifiers. I also wonder if overfitting was a problem throughout as the test scores for Support Vector and Lasso for 0.05 and 0.10 were lower than their respective train scores.

Additionally, I think another possible issue I may have ran into is the size of the dataset. There was 178 datapoints spread out over thirteen features and one target. Would this project have benefitted by a larger dataset? I think in part, why Lasso was so successful is because regularization is a part of its design, so it was able to weed out on its own some of the noise whereas the other ones are not designed to do that. Between the milestone 3 and 4 turn in I had played around with the idea of adding in some dimensionality reduction to the overall dataset but thought that might negate the point of Lasso being included and had ultimately decided to scrap that idea.

After completing this project I wonder, at what point [for each classifier] is a dataset too small? Is there an easier way than trial and error to realize you're overfitting a model? Especially after reading the Overfitting Iceberg in week 8, I'm particularly curious as someone who thinks overfitting may have played a part in my poor scores, if there is even a point in using the classifiers I used in this project if Neural Networks with the application of Deep Learning can completely negate the issues I ran into during this project.

References:

Supervised Machine Learning: A Review of Classification Techniques by SB Kotsiantis

Prediction of Diabetes using Classification Algorithms by Deepti Sisodia and Dilip Singh Sisodia

Comparative analysis of image classification algorithms based on traditional machine learning and deep learning by Pin Wang, En Fan, and Peng Wang

Effectiveness Evaluation of Rule Based Classifiers for the Classification of Iris Data Set by C. Lakshmi Devasena, T. Sumathi, V.V. Gormathi, and M. Hermalatha

Analysis of white wine using machine learning algorithms by Manisha Koranga, Richa Pandey, Mayurika Joshi, Manish Kumar

The Overfitting Iceberg by Chenlei Fang, Jerry Ding, Qicheng Huang, Tian Tong and Yijie Sun