Auto-Sklearn Fail

Finn Tomasula Martin

COSC-4557

---

1 Introduction

One of the things that makes machine learning difficult is the lack of knowledge you have about the learning process. Many ML techniques are by definition black-box problems, meaning all you need to worry about is what you feed into the process and what comes out. In general, this can save you a lot of work and effort but it also means that if things to not turn out how you expect them to, it can be difficult to figure out why. One issue that commonly occurs is that optimized models end up having worse performance than their base counter parts. This is an issue which could be caused by any number of things and can be difficult to resolve. So, in this report, we will take a look at an example of this exact issue and see if we can fix it.

---

2 Dataset Description

The dataset we are using today measures the quality of various wines based on 11 feature variables. It contains 1599 observations with no missing values. The dataset classifies these wines into a scale of quality from 3-8 (3 being the worst, 8 being the best).

---

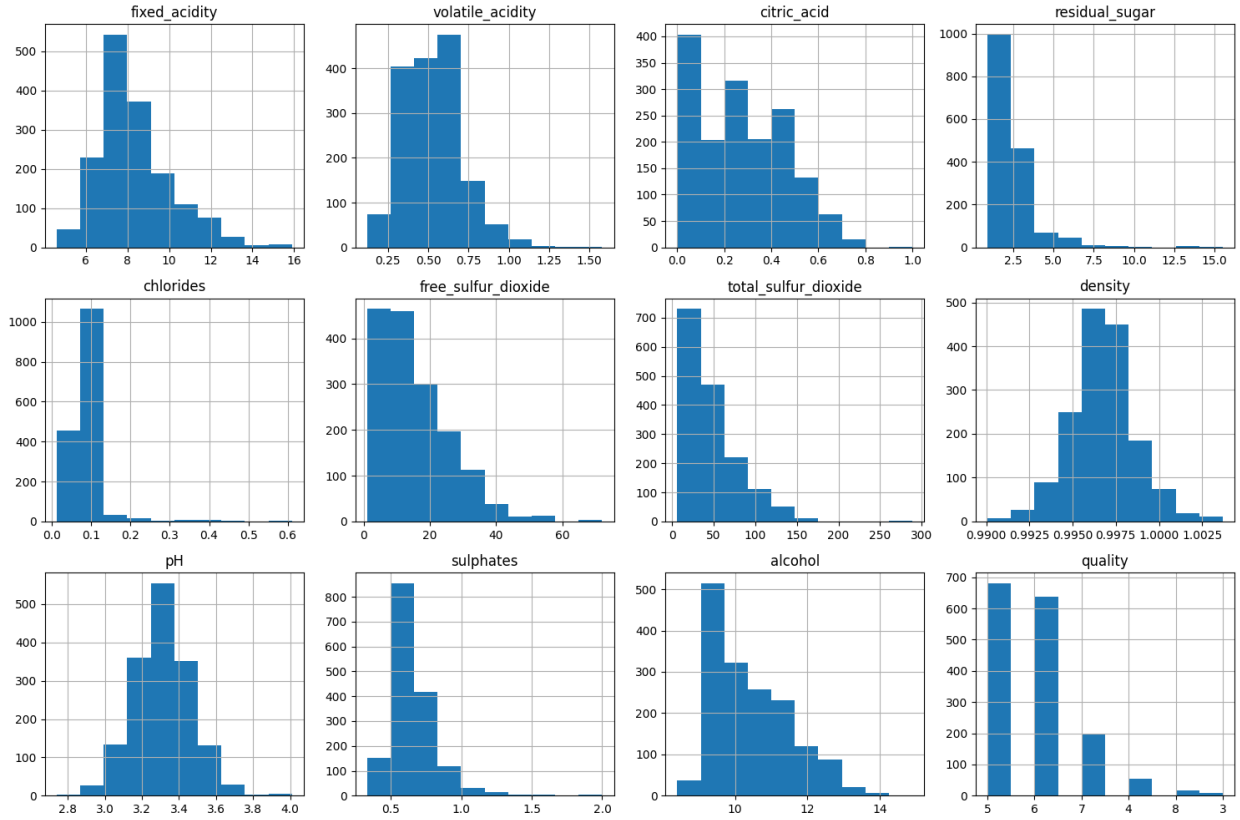
3 Results of Analysis

For this analysis we will be using the auto-sklearn framework of python. We will run a base random forest and then optimize it for five minutes and compare performance. This appears to be rather straight forward but if you run that analysis as described you will see that the optimized model actually performs worse than the base model. Specifically we get the following classification accuracy for each model:

Model	Classification Accuracy
Base	0.67
Optimized	0.6425

So, now the question is, why is this happening and how can we fix it?

To start, lets take a look at the visualization of all our variables to see if anything looks like it may be causing issues.



One of the first things that may stick out is that features do not appear to have a uniform distribution. We can fix this by normalizing them. After normalizing, we get the following results:

Model	Classification Accuracy
Base	0.6775
Optimized	0.655

Unfortunately, this did not fix our issue but it did improve the performance of both models so we might as well keep it.

The next thing we can try is changing our method of nested resampling. By default sklearn resamples by splitting the data into a 66.7% train set and a 33.3% test split. It is possible that this split is adding some sort of bias to our optimization process and causing it to not find an optimal model. To fix this possibility we can switch to 10-fold cross validation for the inner resampling. This will cause sklearn to try every configuration with different folds as the test set and choose the best average configuration. So, it should remove any bias introduced by a bad train test split. After making this change we get the following results:

Model	Classification Accuracy
Base	0.6775
Optimized	0.675

Again our problem is not solved but this time our optimized model is much closer to the base one. That may mean that our problem has something to do with how the data is organized.

Earlier, we normalized all our feature variables but we didn't touch our target variable. If you refer to the figures above, you may notice that the classes of the target variable, quality are very imbalanced. There are over 600 observations of quality 5 and 6 while the rest do not have nearly as many. This again may be introducing bias into our model as it will be really good at predicting wines of quality 5 or 6 but nothing else. That means for any observation it is more likely to predict a quality of 5 or 6 even if that is not the most accurate classification. There are several methods for dealing with the imbalance of the target variables classes including, oversampling, undersampling and adding weights to the under represented classes. For this analysis we will use stratification. Stratification works by splitting our outer sampling into splits that maintain the same class imbalance. After stratification we get the following results:

Model	Classification Accuracy
Base	0.6775
Optimized	0.69375

And just like that we now see better performance in our optimized model! That means the issue was likely being caused by the imbalanced data. In conclusion, issues in ML can be fairly difficult to diagnose but through knowledge of the techniques/technology and some experimentation, we can usually get to the bottom of the issue.

---

5 code

See fail.py for failure code

https://github.com/COSC5557/auto-sklearn-fail-ftomasul/blob/main/fail.py

See fix.py for successful code

https://github.com/COSC5557/auto-sklearn-fail-ftomasul/blob/main/fix.py

---

Sources

“6.3. Preprocessing Data.” *Scikit*, scikit-learn.org/stable/modules/preprocessing.html. Accessed 16 Apr. 2024.

“Apis.” *APIs - AutoSklearn 0.15.0 Documentation*, automl.github.io/auto-sklearn/master/api.html. Accessed 16 Apr. 2024.

DSH, Team. “Handling Imbalanced Datasets in Scikit-Learn: Techniques and Best Practices.” *Data Science Horizons*, 21 July 2023, datasciencehorizons.com/handling-imbalanced-datasets-in-scikit-learn-techniques-and-best-practices/.

ChatGPT-3: Can you show me how to visualize the features of a dataframe in python?