Wine Prediction Report

Author:
Tianze Liu(tliu76@binghamton.edu)
Xianzhi Luo(xluo22@binghamton.edu)

Introduction:
Wine is a popular drink, at large banquets or family parties, wine is always the first choice for most people. Nowadays wine is increasingly enjoyed by a wider range of consumers and there are a variety of wines to choose from for consumers, therefore, it is important to know how to evaluate a bottle of wine by its quality.

In this project, we want to train a model with a dataset of white wines (from Portugal) and use the model to predict the quality of a wine. We get the data from UCI ML Repository and we decide to use random forest(rf) to do the training because we've learned this algorithm in our presentation this semester and also rf has a good performance on this dataset.

Technical Approach:
To solve the classification problem, we planned to use SVM as our tools. But, after the presentation in the midterm, the professor gave us some advice on how to improve our project(poor accuracy is about 62%). One is that SVM is not very suitable for our problem, maybe we can choose other methods. Another is decreasing the number of categories of the label(the label supposedly ranges from 0 to 10).

Our dataset is with 11 attributes(physicochemical contents) and 1 label(wine quality).it has no missing value but is very imbalanced. Also, the relevance among these features is not clear.
Considering the specialty of our dataset, we choose random forest as our algorithm. According to our reference, random forest is very good at handling high-dimension and unbalanced data, which is seemingly perfect for our problem. Simply put, a random forest consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction.

As for the experimental procedure, firstly, we roughly measure the accuracy of our algorithm in a multi-classification case and found the accuracy is nearly 66%, very poor and unacceptable for us.

Therefore, we change our track to a classification problem with fewer labels. In our case, we check and show the imbalance of the original dataset via a pie chart.

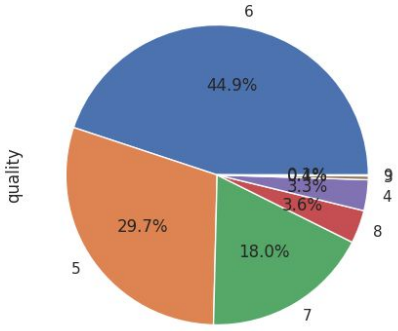

Figure 1. “Quality” label before preprocessing

Then, we turn this problem into a ternary one(the label becomes “low(<=5)” “medium(>=5 and <=7)” “high(>7)”). By doing this, some missing labels are masked(e.g. There doesn’t exist a data point with labels 0 - 2 and 10), and the requirement of the precision of our solution becomes less strict.

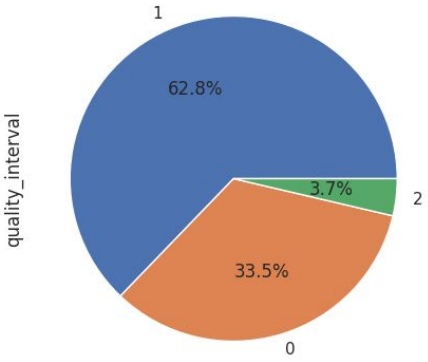

Figure 2. “Quality” label after preprocessing

After this, we tried to discuss the feature selection, we plotted a heatmap to show the correlation among these features and found there don’t exist strictly correlated feature groups. Thus, we keep all the features. The next part is data scaling, and then the train and test dataset gets split from the original dataset. The preprocessing part ends.

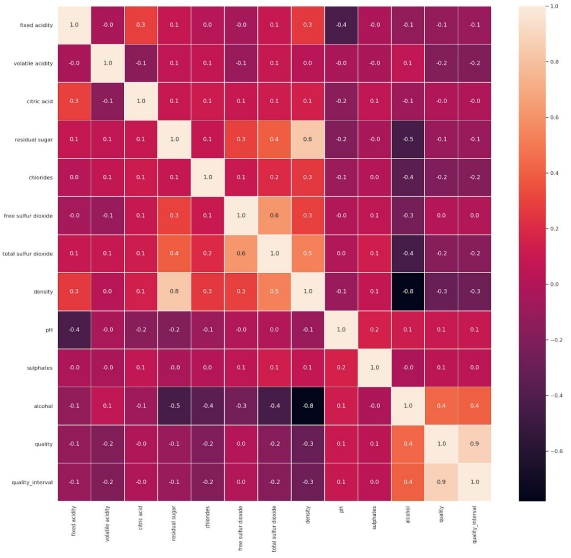

Figure 3: Correlation matrix. From this picture we can see there are no two different physicochemical features(the first 11 features) sharing 1.0(positive) or -1.0(negative) correlation coefficient, which means there don't exist two features that are totally related with each other. That's why we keep all the features

Next, we try to fine-tune the hyperparameters in our model. In our case, we only consider two hyperparameters of the random forest model, the number of trees and the considered number of features at each split when building each tree. After coupled with the cross-validation method, we get the best combinations under which our model gets the best measurement score. To be noted, due to limited computation resources, we only consider 10 combinations(number of trees are 1000 1500 2000 2500 3000 and the number of considered features are "sqrt" or "log"). Then we use GridsearchCV to do the hyperparameter tuning.

Finally, we make sure of the number of selected features and the best combination of hyperparameters. Then, we fit the test set into the model and get the final accuracy.

Results and Discussion:

Our final result on binary class labels is much better than that on multiple class labels that the former has about 10 percent accuracy higher than the latter. We classify the quality of wine into low medium or high instead of giving a rating from 0 to 10 on each kind of wine. Our training on binary classification will achieve a 0.817 accuracy with the hyperparameter "n_estimators" = 2000, which means our model's tree number is 2000, and the hyperparameter "max_features"= "sqrt" which means at each node we split an sub-feature which contains sqrt number of the total attributes.

```
/home/likle/PycharmProjects/test2/venv/bin/python /home/likle/Desktop/580ml/Wine_quality/wine_prediction.py
Best parameters:
{'max_features': 'sqrt', 'n_estimators': 2000}
Mean accuracy
0.8173469387755102

Process finished with exit code 0
```

Figure 4: Result

As mentioned above, we only consider 10 combinations of hyperparameters, so this result may not be the best model we can get if we increase the number of trees to [1000,10000].

Random forest induces low overfitting because of its bootstrap and random feature-picking strategy and after we reclassify our dataset to only three classes:"low", "medium" and "high", outliers are eliminated(e.g. Very few data points with quality 9, non-existing data points with label 1 or 2). Thus this result is more reliable. And obviously it has much better performance than rf on class label [3,9].

Conclusions and Future Work

In this project, we apply new knowledge we've gained this semester into practice. Although our implementation still has many weaknesses due to various reasons, we still learned many things from this project.

In preprocessing, we found this dataset is unbalanced and if we just apply rf to it we will get poor performance, then we learned to reclassify the class label to make our dataset more balanced. The rf works better on a more balanced dataset which isn't affected by missing class labels: [0,2] and 10 after we do the reclassifying.

We also applied feature selection knowledge which is calculating the correlation matrix for all the features of this dataset and found we need to keep all these features because there are no two features that are totally related with each other.

For future work, actually we consider a lot. More algorithms can be applied to do the training and testing, such as KNN, SVM, even, a single decision tree and each of them may have different performance on this dataset and we can make comparison and get more insights on machine learining algorithms. Also more hyperparameters in random forests can be considered such as max_depth, min_sample_split, min_sample_leaf and so on. That may end up with a better and more precise random forest model.

Reference:

[1]https://scikit-learn.org/stable/modules/cross_validation.html
[2]https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
[3]https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
[4]https://archive.ics.uci.edu/ml/datasets/Wine+Quality
[5]https://towardsdatascience.com/why-random-forest-is-my-favorite-machine-learning-model-b97651fa3706
[6]https://seaborn.pydata.org/generated/seaborn.heatmap.html
[7]https://scikit-learn.org/stable/modules/feature_selection.html