COSC 5557
Machine Learning - Algorithm Selection

Almountassir Bellah Aljazwe

March 2024

1 Introduction

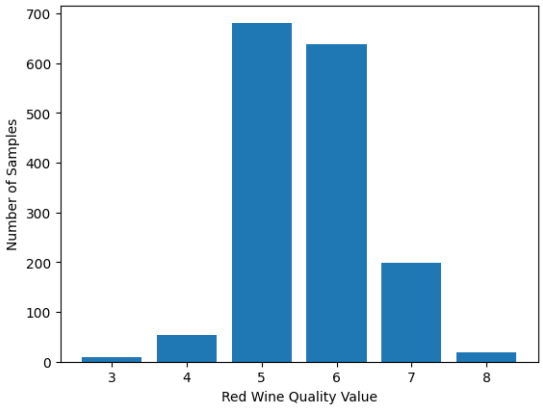
Our initial attempt at creating an effective machine learning model, for the 'Red Wine Quality' dataset, proved to be a massive failure (see the 'Warm-up' activity results). As a result, our main objective, in this report, is to widen our search for a model that offers a solid foundation for an effective predictive model for our dataset. The method of search will be to pick five classification models, with fixed hyper-parameters that can be set by us or provided by Scikit-Learn through their default model initializations. After training and making predictions with our chosen classification models, we conclude by evaluating our models using three different evaluation methods; based on these test results, we then attempt to choose the best-performing model.

2 The Dataset

The 'Red Wine Quality' dataset contains 1599 rows (samples) and twelve columns; eleven are feature columns, while the final twelfth is the target column. The target column contains categorical values; each value is an integer which corresponds to a 'quality' ranking from '1' to '10'. The dataset, however, ranges from '3' to '8' only, with all six values appearing. The dataset contains no missing values for all the provided samples.

A defining characteristic of the 'Red Wine Quality' dataset is its imbalanced distribution of target values. This can be concluded from the following :

3	4	5	6	7	8
10/1559	53/1559	681/1559	638/1559	199/1559	18/1559
$\approx 0.6\%$	$\approx 3\%$	$\approx 44\%$	$\approx 41\%$	$\approx 13\%$	$\approx 1\%$



From the figures, we can clearly observe that the data distribution is cluttered around the middle target values. This, in reality, is reasonable as it may be rare to encounter wine with extremely low quality or extremely high quality; it is reasonable to expect wine with average quality. With regards to our model, the imbalanced nature of this dataset may present obstacles in the performance of our predictions; this is due to the low number of samples for quality values, other than '5' and '6'.

3 Technologies Used

The main technologies used, throughout this process, are the Python programming language and the powerful Scikit-Learn library, which is built on top of Python. To produce the figures, Seaborn, which is also a library built on top of Python, is used.

4 Chosen Classification Models

As mentioned, five classification models were chosen as potential candidates for the development of an effective predictive model. The following is a list of the five chosen classification models and their main hyper-parameters :

- Logistic Regression
  - Solver - 'liblinear' (Linear classifier)
  - Loss Function - 'l2' (Mean Squared Error)
  - Regularization Strength - 1.0
- K-Nearest Neighbors
  - K-Neighbors - 5
- Random Forest
  - Number of Trees - 100
  - Maximum Depth - 5
  - Criterion - 'gini'
- Support Vector Classifier
  - Regularization Strength - 1.0
  - Degree - 3
  - Tolerance - 0.001
- Decision Tree Classifier
  - Criterion - 'gini'
  - Max Depth - 5

5 Evaluation

5.1 Baseline

An important piece of information to have, before evaluating the chosen models, is a baseline accuracy score for comparison. In the case of a classification prediction problem such as ours, we will consider our baseline the accuracy of simply guessing the most frequently-occurring target value, from our dataset; the most frequently-occurring target value is '5'. If we do the math, our accuracy score will be :

$$\frac{681}{1599} \approx 43\%$$

In other words, our trained models are expected to perform better than the baseline accuracy score of 43%. If not, then that is an indication of something severely wrong in the training of the models.

5.2 Results

The evaluation process for our five chosen models involve three separate evaluation methods. Three different methods are chosen to compare between the results and to ensure no evaluation bias exists when comparing models and their performances. The three evaluation methods are the following :

- A single train-test split accuracy evaluation
- Three independent instances of K-Fold Cross Validation for $K = \{5, 10, 15\}$
- Leave-One-Out Cross Validation

5.2.1 Single Train-Test Split

For this evaluation method, the following table summarizes the accuracy scores, rounded to the nearest integer, for each model :

Logistic Regression	KNN	Random Forest	SVC	Decision Tree
58%	49%	63%	49%	60%

5.2.2 K-Fold Cross Validation

For this evaluation method, the following table summarizes the accuracy scores for each K-fold, rounded to the nearest integer, for each model :

K	Logistic Regression	KNN	Random Forest	SVC	Decision Tree
5	57%	44%	59%	50%	55%
10	58%	45%	59%	50%	56%
15	58%	44%	60%	50%	54%

5.2.3 Leave-One-Out Cross Validation

Since this evaluation method, trains on all but one random sample from the dataset, the total number of evaluations is the number of samples in the dataset : 1599. As it wouldn't be useful to place all 1599 evaluation scores, an average is instead taken of all 1599 scores. Thus, for each model, the average, rounded to the nearest integer, is instead provided in the table below :

Logistic Regression	KNN	Random Forest	SVC	Decision Tree
58%	54%	57%	55%	56%

5.3 Conclusion

Our evaluations provide us with some interesting results.

Firstly, we notice that our K-Fold Cross Validation model scores have very low variance and changes across the different K values. The KNN model is the lowest in terms of performance, while the Random Forest and Logistic Regression models are on par.

When looking at the other evaluation methods, however, we observe interesting results. With the single train-test split, we notice increased performance with the KNN, Random Forest, and Decision Tree models; the other two are unchanged relative to the K-fold cross validation results. With the leave-one-out cross validation, we notice that the model performance results are mostly on the same level. Interestingly, the Logistic Regression model is still unchanged, in terms of performance results.

As our evaluation results have shown, the task of deciding which model performed "best" is not so simple. We, first, exclude our single train-test-split evaluation results as that is a biased evaluation method. Focusing only on our K-fold cross validation and leave-one-out cross validation results, we observe that the Logistic Regression, Random Forest, and Decision Tree models produce the best performance results. Picking from the mentioned three, I

believe that the extremely consistent results of the Logistic Regression model may indicate its ability to perform better, with additional hyper-parameter tuning; as a result, the Logistic Regression model looks like the best choice out of the five classification models.