COSC 4010 Practical Machine Learning Fall 2023
Algorithm Selection/Hyperparameter Optimization Report

Derek Walton

October 2023

1 Introduction

In an effort to acquire an understanding of practical machine-learning techniques including algorithm selection and hyperparameter optimization, a series of exercises created by Dr. Kotthoff was developed to allow for hands-on experience applying these techniques. Applying these techniques may significantly impact the performance and efficiency of learners on a specific dataset. The dataset for these exercises is the Wine Quality Dataset from the University of California Irvine Machine Learning repository. A single Python file "algorithm_selection" combines the exercises for algorithm selection and hyperparameter optimization.

2 Dataset Description

The wine quality dataset consists of tabular data. There are two datasets included one for white wine and the other for red, both datasets consist of wine samples from northern Portugal. The datasets consist of 11 features, these features are continuous values for various wine characteristics. The target data are discrete values used to describe the quality of the wine. This repository makes use of the white wine dataset which consists of 4898 observations. The data was divided into a training and test split, 80% of the data was used for training and the remaining 20% for testing.

3.1 Experiment Setup

All learners used in this exercise are implemented in the Python programming language using the Scikit learn library. Pandas and numpy were also used for reading in the data and setting ranges for some hyperparameters. Learners used for these exercises include Random Forest, Linear Regression, Logistic Regression, K Nearest Neighbors, and Decision Tree; a Linear Support Vector Machine was initially included, however, attempts to get the learner to converge were unsuccessful so it was removed. The majority of the algorithms implemented in this exercise make use of a regression algorithm for prediction with the only exception being logistic regression which makes use of a classification algorithm instead. [Scikit-Learn, 2023] The decision to use regression algorithms in the majority of learners was due to the fact that the target values are numeric. In addition, the feature data consisted of continuous values, which made the regression algorithms the most suitable choice for the exercises. The accuracy of all the learners used in this exercise was measured and compared using the *score* function from Scikit-Learn which utilizes an R2 score. This score quantifies how much correlation there is between the feature and target data, i.e., the effect of a change in a value within the feature data on the value of the corresponding target data. [Starmer, 2022] Accuracy was measured on both testing and training data, however, only the accuracy score for testing data was used to compare performance between the learners.

The hyperparameter optimization portion (HPO) of the “algorithm_selection” file makes use of two generic HPO algorithms, grid search and random search. It should be noted that a Bayesian optimization algorithm was implemented using the Scikit-Optimize library, however, long execution times relative to the other algorithms deemed it a poor choice for the hyperparameter search spaces of the learners used in this exercise. The selection of hyperparameter search spaces was based on the parameter description of the learners within their respective documentation, in addition to code examples provided by Scikit-learn developers on the documentation pages for these learners. The decision to implement grid and random search was to observe the performance difference between these two basic HPO algorithms. Each learner used in this exercise employs an unoptimized instance with default values for all hyperparameters. This is done to facilitate a meaningful comparison between the learner instances with hyperparameter optimization (HPO) and those without it. The number of cross-validation for both grid and random search was set to 5 for two reasons. The first is the size of the white wine dataset, many sources suggest using 10 fold however this would leave <500 instances to be used within the validation subset of the data. While this is by no means an insignificant amount, it does not guarantee that the validation set accurately represents the underlying distribution of the dataset as a whole. However, utilizing 5 folds provides the validation set with close to 1,000 instances which is much more likely to accurately represent the underlying distribution of the dataset as a whole. The second reason for choosing a 5-fold cross-validation approach is to minimize the execution time required. The file for this exercise currently takes about half an hour to compile, and it doesn't appear to be worth the additional computation time to increase the number of folds. However, there has yet to be any experimentation with more folds to observe any improvement in learner prediction accuracy.

3.2 Hyperparameter Ranges

This section will discuss the hyperparameters that were tuned for each learner used in this exercise in addition to the ranges used for tuning.

Key: [Sklearn _, 2023], _ is associated with the number next to the Scikit-Learn citation on the reference page.

Learner	**Hyperparameter**	**Description and Range Used**
Random Forest	n_estimators	The number of trees in the forest. [Scikit-Learn 2, 2023] For this exercise, the search space utilized a range form 10-100.
	max_depth	The maximum depth of the tree. [Scikit-Learn 2, 2023] For this exercise, the search space contains the maximum depth of 2, 5, and none was used.
	max_features	The number of features to consider when looking for the best split. [Scikit-Learn 2, 2023] For this exercise, the search space contains the values sqrt and log2.
	bootstrap	Whether bootstrap samples are used then

Learner	Hyperparameter	Description and Range Used
		building trees. [Scikit-Learn 2, 2023] For this exercise, the search space includes the option for True and False.
	min_sample_leaf	The minimum number of samples required to be at a leaf node. [Scikit-Learn 2, 2023] For this exercise, the search space contains the values 2 and 5.
Linear Regression	fit_intercept	Whether to calculate the intercept for this model. [Scikit-Learn 3, 2023] For this exercise, the search space contains the values True and False.
Logistic regression	fit_intercept	See the above row for details.
	solver	Algorithm to use in the optimization problem. [Scikit-Learn 4, 2023] For this exercise, the search space contains all available options for the algorithm. This includes lgfgs (Limited-memory Broyden–Fletcher–Goldfarb–Shanno), liblinear, newton-cg (conjugate gradient), newton-cholesky, sag (Stochastic Average Gradient), saga (variant of sag). [Yahya, 2018].
KNN	n_neighbors	Number of neighbors to use by default for kneighbors queries. [Scikit-Learn 5, 2023] For this exercise, the search space used a range from 5-30.
	weights	Weight function used in prediction. [Scikit-Learn 5, 2023] For this exercise, the search space used the values uniform and distance.
	algorithm	Algorithm used to compute the nearest neighbors. [Scikit-Learn 5, 2023] For this exercise, the search space used the values auto, ball_tree, kd_tree, and brute.
Decision Tree	criterion	The function to measure the quality of a split.

Learner	Hyperparameter	Description and Range Used
		[Scikit-Learn 6, 2023] For this exercise, the search space used the values squared_error, friedman_mse, absolute_error, and poisson.
	spliter	The strategy used to choose the split at each node. [Scikit-Learn 6, 2023] For this exercise, the search space used the values best and random.
	min_samples_split	The minimum number of samples required to split an internal node. [Scikit-Learn 6, 2023] For this exercise, the search space used the values 2 and 5.

4 Results

Figure 1 shows the results of applying the learners used for this exercise with and without optimized hyperparameters. The learner with the best overall performance was the random forest, followed by logistic regression, and KNN.

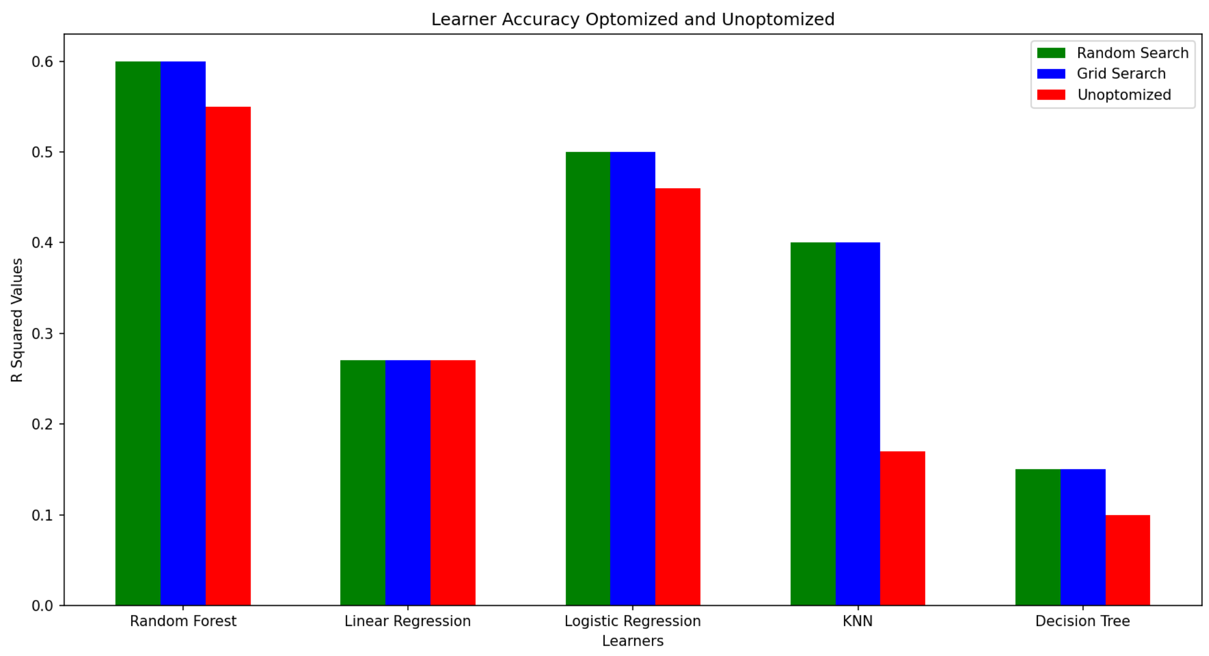
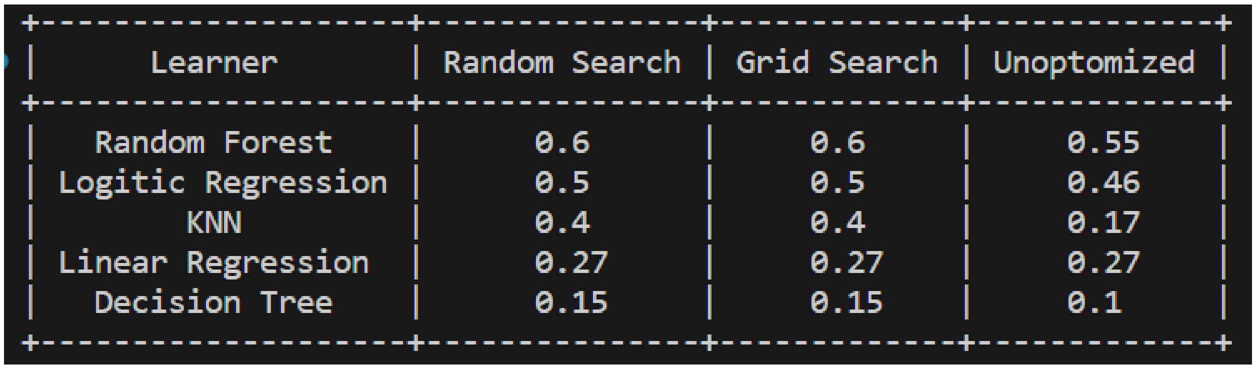


Figure 1: The overall performance of the learners showed a minimal difference between the optimized and unoptimized instances, with the exceptions being KNN and to some extent the Decision tree.



```
+--------------------+--------------+-------------+-------------+
|      Learner       | Random Search | Grid Search | Unoptomized |
+--------------------+--------------+-------------+-------------+
|   Random Forest    |     0.6      |     0.6     |    0.55     |
| Logitic Regression |     0.5      |     0.5     |    0.46     |
|        KNN         |     0.4      |     0.4     |    0.17     |
| Linear Regression  |     0.27     |     0.27    |    0.27     |
|   Decision Tree    |     0.15     |     0.15    |    0.1      |
+--------------------+--------------+-------------+-------------+
```

Figure 2: Table with R2 Scores rounded up to 2 significant figures

5 Code

The associated code can be found in algorithm_selection.py.

References

1). scikit-learn developers. (2023). *1.1. Linear Models*. scikit. https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

Brownlee, J. (2019, May 21). *Difference between classification and regression in machine learning*. MachineLearningMastery.com. https://machinelearningmastery.com/classification-versus-regression-in-machine-learning/

GeeksforGeeks. (2023, April 14). *Classification vs regression in machine learning*. GeeksforGeeks. https://www.geeksforgeeks.org/ml-classification-vs-regression/

Starmer, J. (2022, November 18). *R-squared, clearly explained!!!*. YouTube. https://www.youtube.com/watch?v=bMccdk8EdGo&t=306s

JahKnowsJahKnows 8. (2018, February 22). *How to calculate The fold number (k-fold) in cross validation?*. Data Science Stack Exchange. https://datascience.stackexchange.com/questions/28158/how-to-calculate-the-fold-number-k-fold-in-cross-validation

Naik, K. (2020, October 28). Random Forest hyperparameter tuning using RANDOMISEDSEARCHCV | Machine learning tutorial. YouTube. https://www.youtube.com/watch?v=SctFnD_puQI

2). scikit-learn developers. (2023). Sklearn.ensemble.randomforestregressor. scikit. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html#sklearn.ensemble.RandomForestRegressor

3). scikit-learn developers. (2023). Sklearn.linear_model.linearregression. scikit. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html#sklearn.linear_model.Lhttps://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html#sklearn.linear_model.LinearRegressioninearRegression

4). scikit-learn developers. (2023). Sklearn.linear_model.logisticregression. scikit. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html#sklearn.linear_model.LogisticRegression

Yahya (2018, September 18). *Logistic regression python solvers' definitions*. Stack Overflow. https://stackoverflow.com/questions/38640109/logistic-regression-python-solvers-definitions

5). scikit-learn developers. (2023). Sklearn.neighbors.kneighborsregressor. scikit. https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html#sklearn.neighbors.KNeighborsRegressor.kneighbors

6). scikit-learn developers. (2023c). Sklearn.tree.decisiontreeregressor. scikit. https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html#sklearn.https://

scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html#sklearn.tree.Decision TreeRegressortree.DecisionTreeRegressor