**Hyperparameter Tuning**

Tiati Thelen

Department of Computer Information Sciences, Minnesota State University, Mankato

CIS 677: Individual Studies

Dr. Guarionex Salivia

April 21, 2022

**Contents**

Introduction.….…………………………………..………………….…………………………… 3

Grid Search ……………………….…….……………………………….………………………. 4

Random Search ……………………...…..……………………………….………...……………. 5

Implementation and Results …...………...………………………………..……………..………. 5

References ...……………………….…………………………………….………………..……... 9

# Introduction

# A hyperparameter are variables that influence the training process. Importantly, these parameters are not directly related to the training data. For example, in a neural network one must decide how many hidden layers there are and how many nodes to use. Different machine learning models require different constraints, weights, or learning rates to analyze data patterns (Google, n.d.). Therefore, these measures are defined before training, and they specify how the model training will happen. Hyperparameter tuning is finding the best combination of hyperparameters to give the best accuracy (for our case) for the model. An optimal tuple of hyperparameters produce an optimal model to minimize the loss function associated. This process runs multiple trials in a single training job. With each trial execution is completed from the chosen hyperparameters. Generally, the output will result in a summary, and the optimal values recommended to be utilized. There are a handful of approaches to hyperparameter tuning such as Grid Search, Random Search, Bayesian Optimization, Gradient-based Optimization, etc. This project will consider Grid Search and Random Search.

# This paper will cover the classification algorithm of Random Forests and how hyperparameter tuning applies. Random Forests has a variety of hyperparameters which is why it was chosen (Ismiguzel, 2021). The variables analyzed will be *n\_estimators, min\_sample\_leaf, max\_depth, max\_features, and bootstrap*. Most of these features influence the decision trees and their dependencies within the model. Therefore, the goal is to find the best combination to optimize prediction accuracy.

# The dataset consider is classifying glass type based on different features. It is a discrete dataset with 7 glass categories. The dataset was found on Kaggle (UCI Machine Learning, 2017). In conclusion, the implementation will aim to create the best accuracy score of glass categorization through the hyperparameter tuning of Random Forests. Most information was gathered through the textbook *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems* (Géron, 2019).

**Grid Search**

Grid search is considered the traditional way. It is simply an exhaustive search through the specified hyperparameter spaced defined by the user (Great Learning Team, 2022). The search is assisted by the performance metric, accuracy in this project. It is typically measured through cross-validation. During the training, the process divides the training data further into two parts, the training data and validation data. Including cross validation will be time costly. GridSearchCV takes four arguments: *estimator, param\_grid, scoring, cv*. The *estimator* is the Scikit-learn model, the *param\_grid* is the dictionary with parameter names as keys, the *scoring* is the performance measure (accuracy), and the *cv* is the integer for the number of k-folds for k-fold cross-validation. A downfall to Grid Search is that it is negatively impacted by dimensionality. In return the library will produce the *variable* (GridSearchCV), *variable.best\_params\_*, and *variable.best\_score\_*. It can be time consuming and expensive if the number of hyperparameters is large.

**Random Search**

Random Search replaces exhaustive enumeration of all combinations by selecting hyperparameters randomly. It will apply to a discrete setting but will generalize to be continuous. It can output perform Grid Search, especially when there is only a small number of parameters. Random Search has a low intrinsic dimensionality. This means that a small number of hyperparameters will affect the final performance of the algorithm. Also, it will allow the inclusion of prior knowledge by specifying the distribution from the sample. A disadvantage of Random Search is that during computing it can have high variance. This is due to the selection of parameters being completely random. The RandomSearchCV also takes in similar input as the GridSearchCV. The biggest difference between Random Search and Grid Search is that Random Search has knowledge about the distribution or list of hyper parameters. Since it is known, the hyperparameters values are picked up at random from this distribution.

**Implementation and Results**

The dataset selected needed minimal data cleaning. In the figure below, it shows there is not missing values within the dataset. Therefore, implementation of the algorithms can be done right away. The test data set is 20% of the original data.

**Graphical user interface, text

Description automatically generated**

**Figure 1: Data Processing**

A baseline is needed to be determined before hyperparameter tuning can begin. The paper creates a Random Forest model and finds its accuracy score. From Figure 2, the output accuracy is 83.7%. Overall, this is an acceptable score for the model.

**Graphical user interface, text, application, email

Description automatically generated**

**Figure 2: Random Forest Accuracy**

Next the paper will implement the hyperparameter tuning algorithms. Random Search will be the first one. As mentioned earlier, only a select few hyperparameters will be considered. This due to the limited schedule and the smaller dataset. Scikit-learn provides a library for allowing Random Search to be done in Python. Once the search is complete the code will select the optimal combination and remodel on the data. The new accuracy will be output to the screen. Random Search produced a better accuracy of 86% shown in Figure 3.

**Graphical user interface, text

Description automatically generatedGraphical user interface, text, application, email

Description automatically generated**

**Figure 3: Random Search Implementation and Accuracy**

Lastly, Grid Search is implemented using a similar library through Scikit-learn again. The same hyperparameters from Random Search will be considered here. An 83% accuracy was produced with is the same score as the regular Random Forest. Therefore, there is no increase in the accuracy score. Figure 4 below shows the implementation and output.

**Graphical user interface, text

Description automatically generatedGraphical user interface, text, application, email

Description automatically generated**

**Figure 4: Grid Search Implementation and Accuracy**

Putting it all together, Table 1 is created. This is to easily examine all the accuracy scores produced in the project. As mentioned prior, Random Forest and Random Forest with Grid Search produced the same score of 83%. The best output came from Random Forest with Random Search. In conclusion, it is best to use the hyperparameters determining with Random Search for modeling this problem. There are a small about of hyperparameters, so these results are intuitive.

|  |  |
| --- | --- |
| Algorithm | Accuracy |
| Random Forest | 83% |
| Random Forest + Grid Search | 83% |
| Random Forest + Random Search | 86% |

**Table 1: Combined Accuracy Scores**

**References**

Géron A. (2019). *Hands-on machine learning with scikit-learn and TensorFlow concepts, tools, and techniques to build Intelligent Systems*. O'Reilly.

Google. (n.d.). *Overview of hyperparameter tuning*. Google. Retrieved May 2, 2022, from <https://cloud.google.com/ai-platform/training/docs/hyperparameter-tuning-overview>

Great Learning Team. (2022, March 22). *Hyperparameter tuning with GRIDSEARCHCV*. GreatLearning Blog: Free Resources what Matters to shape your Career! Retrieved May 2, 2022, from https://www.mygreatlearning.com/blog/gridsearchcv/

Ismiguzel, I. (2021, September 29). *Hyperparameter tuning with Grid Search and Random Search*. Medium. Retrieved May 2, 2022, from https://towardsdatascience.com.

UCI Machine Learning. (2017, January 27). *Glass classification*. Kaggle. Retrieved May 2, 2022, from https://www.kaggle.com/datasets/uciml/glass

Wikimedia Foundation. (2022, April 9). *Hyperparameter optimization*. Wikipedia. Retrieved May 2, 2022, from https://en.wikipedia.org/wiki/Hyperparameter\_optimization