Comparing Classification Models
Supervised Machine Learning

Anonymous Authors[1]

Abstract

Caruana and Niculescu-Mizil conducted a comprehensive empirical evaluation of supervised learning; in a classification setting, they compared the following supervised learning methods: SVMs, neural nets, logistic regression, naive bayes, memory-based learning, random forests, decision trees, bagged trees, boosted trees, and boosted stumps (Caruana & Niculescu-Mizil, 2006). In 2024, this paper serves to revisit the original paper from 2006 and conduct another comparison between a subset of these supervised machine learning methods. The paper will primarily focus on logistic regressions, XGBoosts, random forests, and decision trees.

1. Introduction

This paper is part of the COGS 118a final project. The purpose of the paper is to analyze the efficacy of various supervised machine learning algorithms in classification tasks and compare their performances with one another.

The GitHub repository link for this project can be found at:

https://github.com/e7song/comparing-classification-algorithms

1.1. Dataset Source

Repository The three datasets used in this project come from the UC Irvine Machine Learning Repository. This mirrors the data source of the original paper. Some of the datasets that are used were also used in the original paper; others are new datasets. These data sets are used in a classification task.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

1.2. Exploratory Data Analysis and Processing

Adult This dataset was donated to the UCI Machine Learning Repository in 1996 with the intent on "predicting whether annual income of an individual exceeds $50K/yr based on census data" (Becker & Kohavi, 1996). The dataset has 48,842 instances with 14 features and a binary label for those who earn more or less than $50k/yr year. The dataset is fairly balanced, with 24720 negative labels and 24122 positive labels. The non-categorical variables had the standard scaling function applied to them while the categorical variables were one-hot encoded. The final dataset shape (including the label) was 48,842 x 112. More information about the preprocessing of this dataset can be found in the repository under ' /data-cleaning/adult-cleaning.ipynb'.

Wine This dataset was donated to the UCI Machine Learning Repository (UCI). The wine data provides detailed features that describe the physical and chemical properties of wine samples. These features are generally used for analyzing and predicting the quality of wine based on objective measurements. This data is on the lighter side, with 1143 observations and 13 features before any processing. To turn this into a classification task, I am assigning a binary label for wines that have a quality $>= 6$ and a quality of < 6. This quality boundary roughly divides the dataset in half. It would be interesting to compare these algorithms on a smaller set of data as compared to the Adult dataset. More information about the preprocessing of this dataset can be found in the repository under ' /data-cleaning/wine-cleaning.ipynb'.

Heart This dataset was donated to the UCI Machine Learning Repository (UCI). This dataset contains clinical and diagnostic data related to heart disease. One of the columns contains a number from 0-4 that describes the presence of heart disease. These features could be used to predict a potential severeity of heart disease, but instead they have been repurposed to describe the presence of heart disease. By selecting 0 to indicate no heart disease and ¿0 to indicate that there is heart disease, the dataset is roughly partitioned into 45% negative and 55% positive samples. More information about the preprocessing of this dataset can be found in the repository under ' /data-cleaning/heart-cleaning.ipynb'. One important thing to note is that this dataset is smaller

than the wine dataset.

By using three datasets of varying sizes, it gives a more thorough analysis of how these various models were perform against each other in different circumstances.

1.3. Algorithms

Logistic Regression

XGBoost

Random Forest

Decision Tree

2. Methods

2.1. Datasets

Each of the datasets represents a binary classification task. As per the exploratory data analysis and processing section, each dataset has either had their features scaled or has had their features one-hot encoded (whichever is more appropriate). More details on the processes overall can be found in the GitHub repository. The overall structure of the data can be found in '~data'; each dataset is represented as a matrix with the last column belonging to the classification label.

2.2. Dataset Partition

There are three partitions of focus in this paper. The model performances are evaluated on a $20/80$, $50/50$ and $80/20$ training and testing split.

2.3. Model Hyperparameters

Logistic Regression The logistic regression model being implemented is from sklearn (sci). The hyperparameter of interest is the regularization strength C, this paper focuses on $C \in (0, 1)$.

XGBoost The XGBoost model being implemented is from XGBoost (Int). The main hyperparameter of interest is the maximum depth.

Random Forest The random forest model being implemented is from sklearn. The main hyperparameter of interest is the maximum depth.

Decision Tree The decision model being implemented is from sklearn. The main hyperparameter of interest is the maximum depth.

2.4. Model Tuning

For each partition of the dataset, the following steps are performed to find optimal hyperparameters. A grid search on the hyperparameters for each model is conducted on the training data with a five-fold cross validation. For each partition and for each model, a heatmap is generated and an accuracy plot is created. This is to get a sense of how the model is performing for each partition and helps to pick the best hyperparameter. Appropriate samples will be displayed for the $80/20$ training/testing split specifically. This is to save space in the report and prevent cluttering; for specifics on each partition feel free to reference the code in the repository. Each notebook in '~code' refers to the training and testing done on each datasset for each mode.

2.5. Metrics for Comparison

The primary metric for gauging performance on the classification task will be accuracy. This is how many correct the model predicts over the predictions in total. The model accuracy for each hyperparameter is ranked by the average over each split. Once the best hyperparameter is determined, the models will be tested on the testing set. The models are then ranked by their accuracy on the testing set.

3. Experiment

The following plots are displayed for an $80/20$ data partition. The final table detailing all the comparisons are done on the $80/20$ data partition.

3.1. Adult Dataset

3.1.1. Logistic Regression

As the table shows below, this supports the idea that as you have more data in your training, your training accuracy will generally go up.

Table 1. Cross validation accuracies for the different data partitions. (Logistic Regression)

Partition	Best C	Best Avg Training Acc
$20/80$	0.333	0.6637
$50/50$	0.112	0.6720
$80/20$	0.223	0.6711

3.1.2. XGBoost

XGBoost seems to be performing better than Logistic Regression.

3.1.3. Random Forest

Random Forest also has a strong performance.

Table 2. Cross validation accuracies for the different data partitions. (XGBoost)

PARTITION	BEST MAX DEPTH	BEST AVG TRAINING ACC
20/80	2	0.6707
50/50	4	0.6844
80/20	3	0.6742

Table 3. Cross validation accuracies for the different data partitions. (Random Forest)

PARTITION	BEST MAX DEPTH	BEST AVG TRAINING ACC
20/80	16	0.6763
50/50	16	0.6804
80/20	20	0.6817

3.1.4. DECISION TREE

Decision trees performed better than logistic regression, but worse than the ensemble methods.

Table 4. Cross validation accuracies for the different data partitions. (Decision Tree)

PARTITION	BEST MAX DEPTH	BEST AVG TRAINING ACC
20/80	7	0.6649
50/50	9	0.6723
80/20	11	0.6761

3.1.5. FINAL COMPARISON FOR THE ADULT DATASET

Table 5. Testing Accuracies for an 80/20 partition.

MODEL	TESTING ACCURACY
LOGISTIC REGRESSION	0.671
XGBOOST	**0.690**
RANDOM FOREST	0.682
DECISION TREE	0.679

3.2. Wine

3.2.1. LOGISTIC REGRESSION

Logistic Regression performed fairly well on this dataset; the negative and positive labels are distributed fairly evenly (about 54% positive labels). Having a performance of around 76% accuracy is much better than a naive model.

Table 6. Cross validation accuracies for the different data partitions. (Logistic Regression)

PARTITION	BEST C	BEST AVG TRAINING ACC
20/80	0.334	0.7464
50/50	0.666	0.7478
80/20	0.446	0.7495

3.2.2. XGBOOST

XGBoost had a surprising performance in the 20/80 split but was clearly overfitting.

Table 7. Cross validation accuracies for the different data partitions. (XGBoost)

PARTITION	BEST MAX DEPTH	BEST AVG TRAINING ACC
20/80	2	0.7766
50/50	5	0.7689
80/20	2	0.7637

3.2.3. Random Forest

Random Forest had an amazing performance compared to the previous two models.

Table 8. Cross validation accuracies for the different data partitions. (Random Forest)

Partition	Best Max Depth	Best Avg Training Acc
20/80	10	0.7808
50/50	5	0.7689
80/20	2	0.7768

3.2.4. Decision Tree

The decision tree performed the worst here on this dataset.

Table 9. Cross validation accuracies for the different data partitions. (Decision Tree)

Partition	Best Max Depth	Best Avg Training Acc
20/80	10	0.6846
50/50	5	0.7146
80/20	2	0.7178

3.2.5. Final Comparison for the Wine Dataset

Table 10. Testing Accuracies for an 80/20 partition.

Model	Testing Accuracy
Logistic Regression	0.764
XGBoost	0.764
Random Forest	**0.795**
Decision Tree	0.760

3.3. Heart

3.3.1. Logistic Regression

Logistic regression performs quite well on this dataset.

Table 11. Cross validation accuracies for the different data partitions. (Logistic Regression)

Partition	Best Max Depth	Best Avg Training Acc
20/80	0.112	0.8818
50/50	0.223	0.8791
80/20	0.112	0.8496

3.3.2. XGBoost

Table 12. Cross validation accuracies for the different data partitions. (XGBoost)

Partition	Best Max Depth	Best Avg Training Acc
20/80	2	0.8470
50/50	6	0.8791
80/20	7	0.8411

3.3.3. Random Forest

Table 13. Cross validation accuracies for the different data partitions. (Random Forest)

Partition	Best Max Depth	Best Avg Training Acc
20/80	3	0.8970
50/50	3	0.8660
80/20	4	0.8412

3.3.4. Decision Tree

This performs the worst.

Table 14. Cross validation accuracies for the different data partitions. (Decision Tree)

Partition	Best Max Depth	Best Avg Training Acc
20/80	1	0.8652
50/50	1	0.7855
80/20	3	0.7576

3.3.5. Final Comparison for the Heart Dataset

Table 15. Testing Accuracies for an 80/20 partition.

Model	Testing Accuracy
Logistic Regression	0.817
XGBoost	0.800
Random Forest	**0.817**
Decision Tree	0.800

4. Conclusion

Out of the three datasets, the ensemble methods performed the best. Although the datasets were all diverse, **random forests** consistently performed well. XGBoost also performed pretty well, closely following or outperforming random forests in certain cases. Even with more careful parameter hypertuning, logistic regression and decision trees fell behind random forests and XGBoosts. Please reference the appendix for specific heatmap relationships between the hyperparameters and the cross-fold validation accuracies. There were several limitations that need to be acknowledged at the end of this paper. There was a significant resource constraint which limited the number of hyperparameters to be tuned. Certain models that XGBoost had more parameters than just depth which were not explored as thoroughly as possible. Furthermore, the heart disease dataset is limited in scope since it was hard to fully acquire all the details about the patients. Further extensions to this work can be greatly simplified in the future, as explained in the bonus section.

5. Bonus Section

The coding notebooks have functions that greatly streamline the training and testing process. This could greatly simplify any future works that build on this paper; more parameters can be added in as long as certain formats are followed with the data.

References

URL https://scikit-learn.org/1.5/modules/generated/sklearn.linear_model.LogisticRegression.html.

Becker, B. and Kohavi, R. Adult. UCI Machine Learning Repository, 1996. DOI: https://doi.org/10.24432/C5XW20.

Caruana, R. and Niculescu-Mizil, A. An empirical comparison of supervised learning algorithms. *Proceedings of the 23rd international conference on Machine learning - ICML '06*, pp. 161–168, 2006. doi: 10.1145/1143844.1143865.

A. Heatmaps and other relevant plots

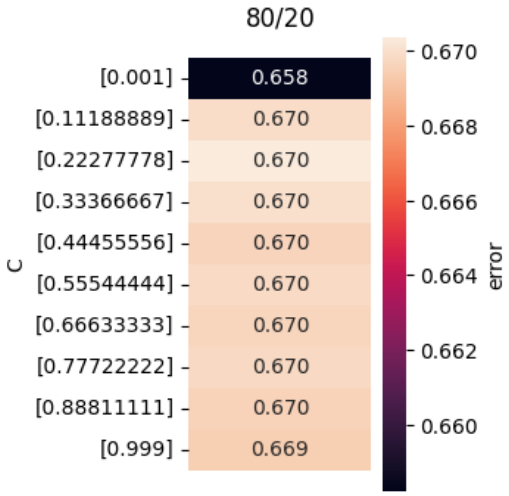

Figure 1. Capturing the average training accuracies during gridsearch for logistic regression. (Adult)

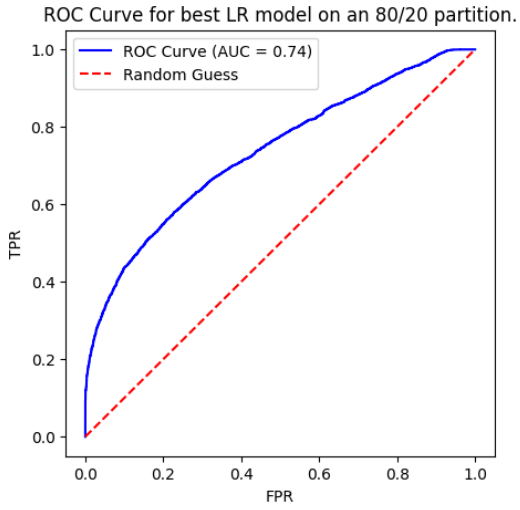

Figure 2. Comparing performance to the baseline. (Adult)

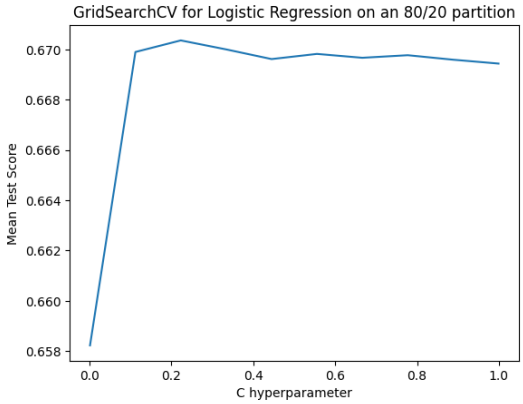

Figure 3. Showing how the accuracy changes with respect to the hyperparameters. (Adult)

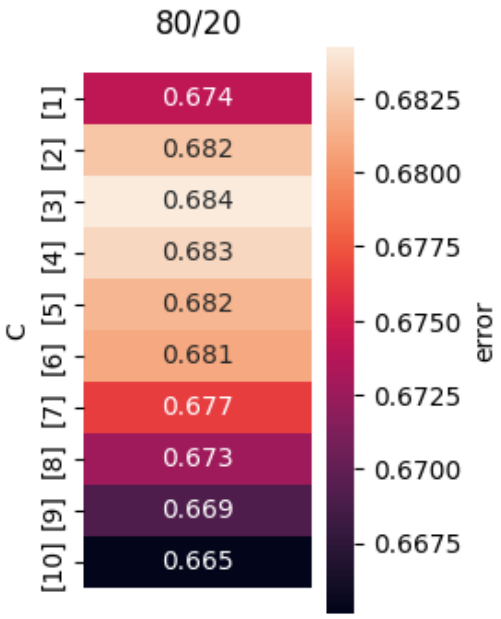

Figure 4. Showing how the accuracy changes with respect to the hyperparameters (Adult, XGBoost, $80/20$ split.)

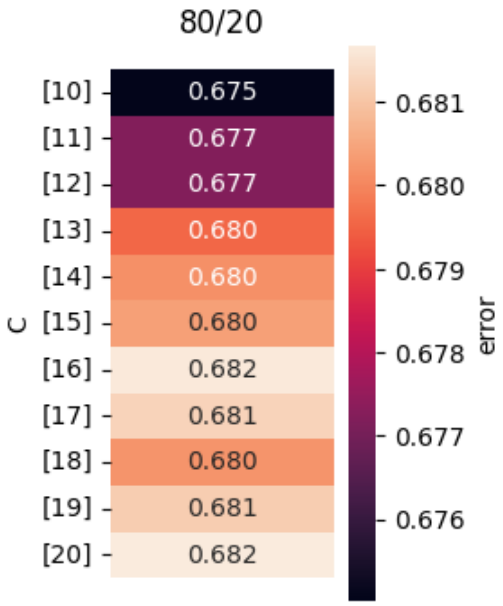

Figure 5. Showing how the accuracy changes with respect to the hyperparameters (Adult, Random Forest, $80/20$ split.)

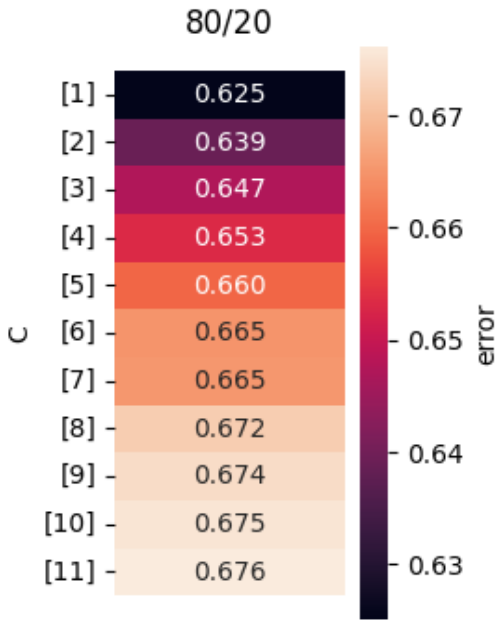

Figure 6. Showing how the accuracy changes with respect to the hyperparameters (Adult, Decision Tree, $80/20$ split.)

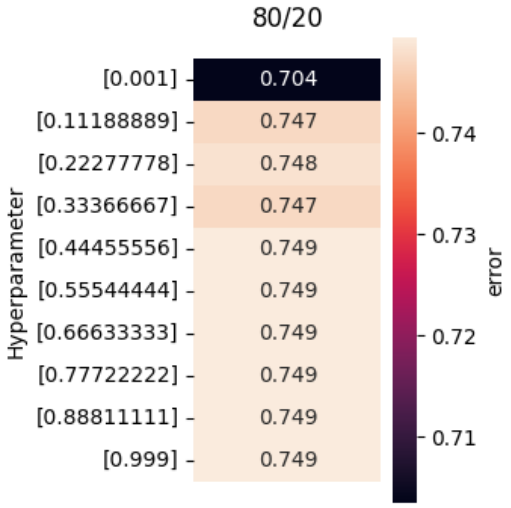

Figure 7. Showing how the accuracy changes with respect to the hyperparameters (Wine, Logistic Regression, $80/20$ split.)

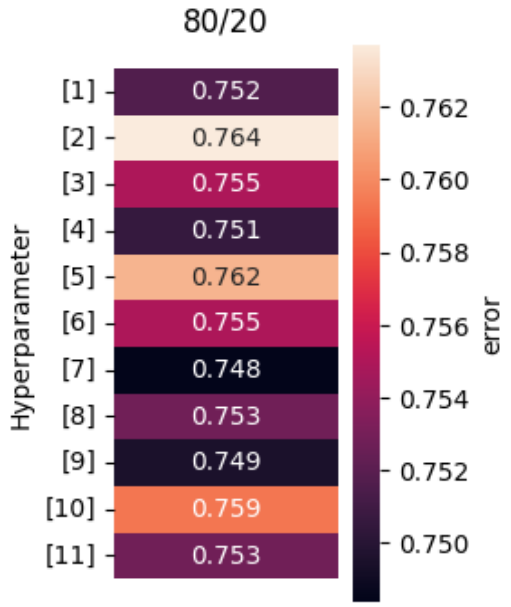

Figure 8. Showing how the accuracy changes with respect to the hyperparameters (Wine, XGBoost, $80/20$ split.)

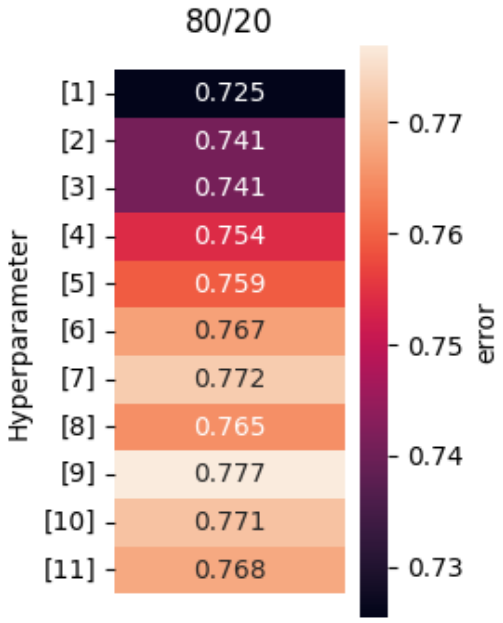

Figure 9. Showing how the accuracy changes with respect to the hyperparameters (Wine, Random Forest, $80/20$ split.)

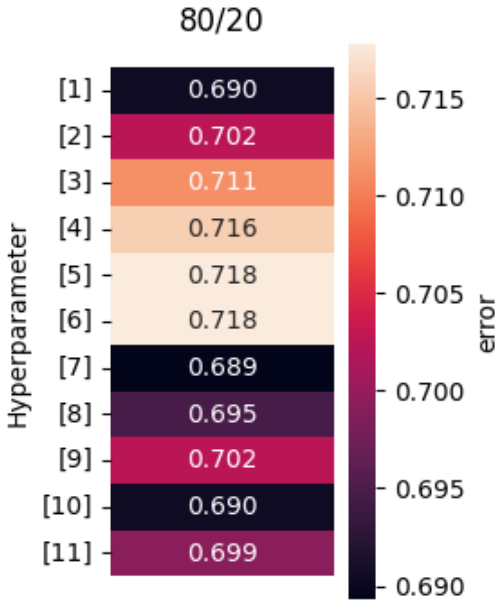

Figure 10. Showing how the accuracy changes with respect to the hyperparameters (Wine, Decision Tree, $80/20$ split.)

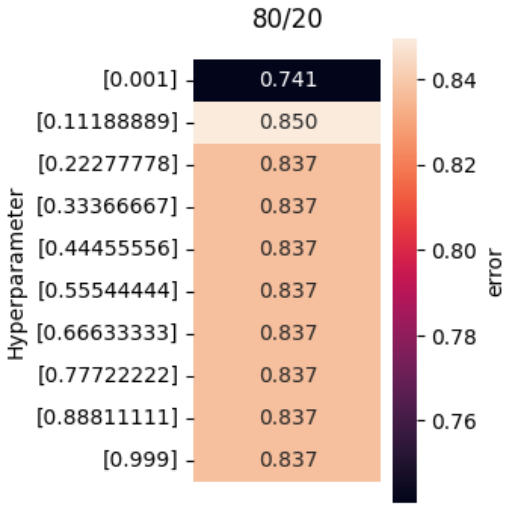

Figure 11. Showing how the accuracy changes with respect to the hyperparameters (Heart, Logistic Regression, $80/20$ split.)

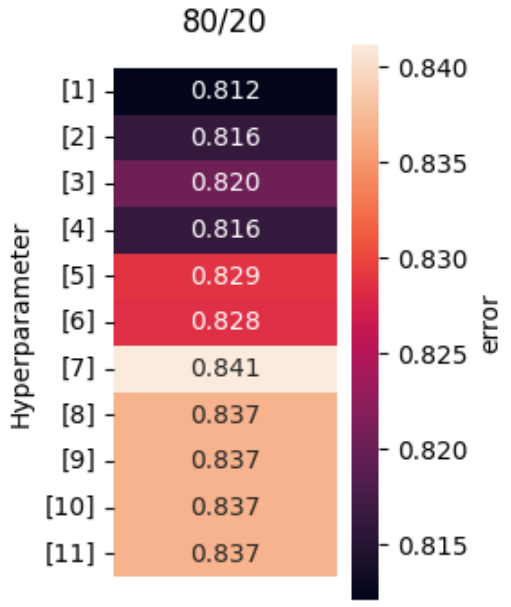

Figure 12. Showing how the accuracy changes with respect to the hyperparameters (Heart, XGBoost, $80/20$ split.)

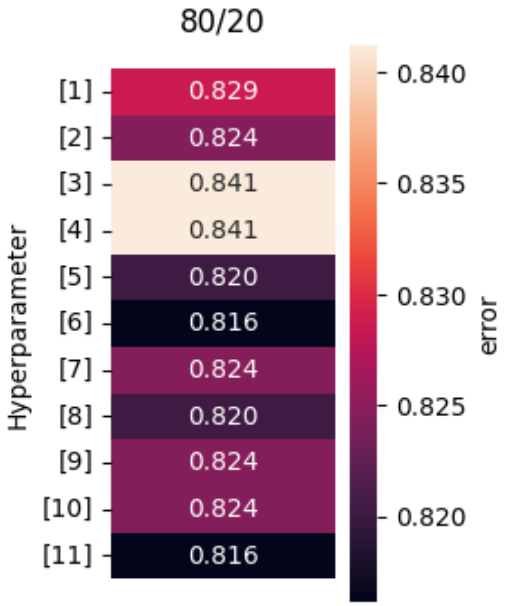

Figure 13. Showing how the accuracy changes with respect to the hyperparameters (Heart, Random Forest, $80/20$ split.)

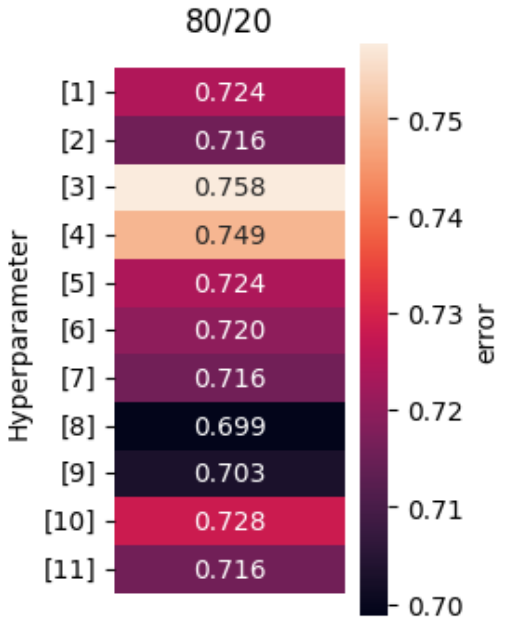

Figure 14. Showing how the accuracy changes with respect to the hyperparameters (Heart, Decision Tree, $80/20$ split.)