**2023-Apr-27**

**BGU**

**Computational Learning**

**Assignment 1**

**Roei Zaady 318747946**

**Omer Yanai 024093866**

This report presents a summary of the results of building a Decision Tree Classifier, and a Bagging Classifier in Python.

A Decision Tree Classifier, is a model which attempts to classify data into groups based on previous observations. It works by building a tree-like model of conditions that will divide the data into groups which have shared attributes. Each node in the tree represents a decision, based on a feature in the input data, and each split represents the outcome of that decision. At the bottom of the tree there are leaves, which represent the final predicted classification. The Decision Tree Classifier is easy to interpret and visualize, making it a popular algorithm in machine learning. In this assignment, all features are assumed to be binary.

A Bagging Classifier (Bootstrap Aggregating), is a model that uses a group of Decision Tree Classifiers to reach a prediction of classification. The Bagging Classifier works by training multiple base classifiers, such as Decision Trees, on different subsets of the training data, and then combining the predictions of these classifiers to make a final prediction. By using an ensemble of Decision Trees in which each tree saw only parts of the data, the Bagging Classifier can help reduce overfitting and improve the generalization of the model.

Our work compares the performance of two bagging classifiers in a binary classification task (target classes are 0/1), on 5 different datasets, using 5 different metrics (accuracy, precision, recall, F1, AUC-ROC).

The Classifiers:

1. MyID3: MyBaggingID3: Our Bagging ensemble classifier using “MyID3” - Our Decision Tree developed from scratch.
2. SKLearn BaggingClassifier using SKLearn’s Decision Tree Classifier as base estimators.

The Datasets ( source and shape) :

1. Tic-Tac-Toe: [UCI](https://archive.ics.uci.edu/ml/machine-learning-databases/tic-tac-toe) (958,27)
2. Breast Cancer Wisconsin: [UCI](https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin) (699, 90)
3. Mushroom: [Kaggle](https://www.kaggle.com/datasets/uciml/mushroom-classification) (8124, 117)
4. Haberman: [UCI](https://archive.ics.uci.edu/ml/machine-learning-databases/haberman) (306, 94)
5. Monk: [UCI](https://archive.ics.uci.edu/ml/machine-learning-databases/monks-problems) (432, 17)

Columns which contain no classification value (e.g. unique identifier per row), or columns with wide numeric range and no significant value to the predictions were removed. All other columns were preprocessed using Pandas’ get\_dummies to “One Hot Encode” columns of binary values.

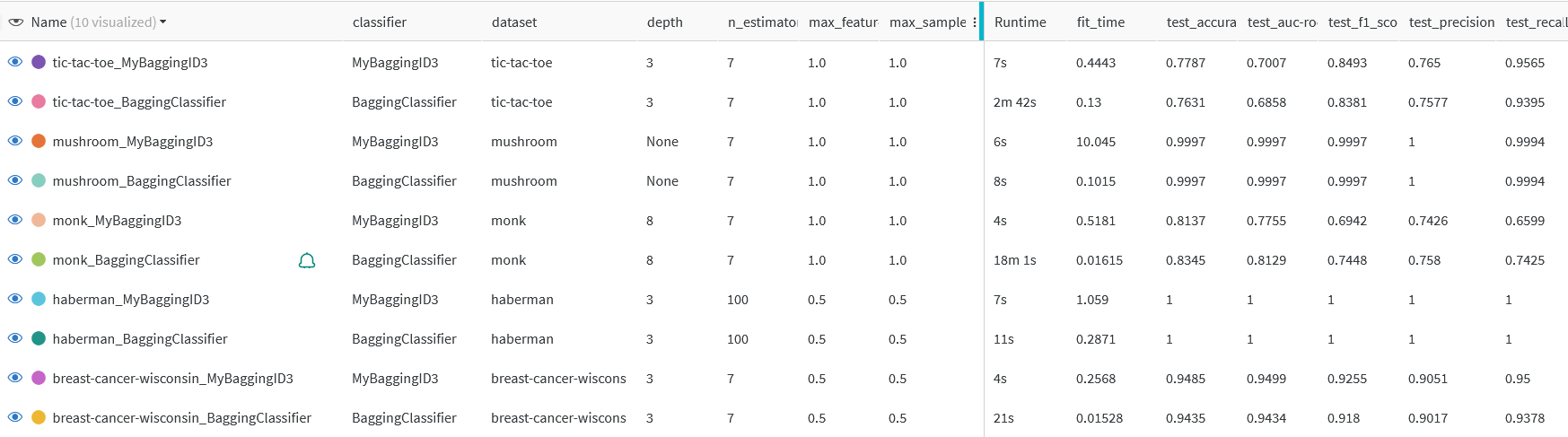
In order to evaluate the generalization capability of the different classifiers, we ran repeated K-Fold cross validation, subdividing the dataset into several folds, averaging the results from the different iterations in order to get a more robust measurement of performance by reducing both variance and bias caused by random data sampling.

Our assumption prior to running the evaluation on the different models, was that both bagging classifiers will perform about the same in all metrics, but that SKlearn’s Bagging Classifier will be faster, as it is better adapted for a more efficient data handling.

During the evaluations of the classifiers and experimentation with the various datasets, we observed that two of the datasets are yielding perfect classification scores when allowed to use unlimited tree depth, therefore a limit (e.g max\_depth=3) was introduced. By making the classification task more challenging, the models’ predictions accuracy was impaired and would therefore allow us to better compare between the different classifiers.

**Evaluation:**

We ran 60 different experiments in Google Colab, using different configurations (defined in an external CSV file “runs.csv”). During the evaluation, we collected the results using the “[Weights & Biases](https://wandb.ai/home)” platform. [Link to table.](https://api.wandb.ai/links/bgu_ml_/x0uf1b6r)



The table below shows a partial snapshot of the collected results. Each run is showcased with the means of each metric, as well as the hyperparameters used in the models.

As can be seen, our “MyBaggingID3” Classifier performs similarly to SKlearn’s “BaggingClassifier” in terms of Accuracy, Precision, Recall, F1 Score and AUC-ROC.

As expected, SKlearn’s Bagging Classifier is better adapted for speed and efficiency.

The full table of configurations and results can be seen in Appendix A.

**Additional Findings from running all 60 configurations:**

1. Comparison of MyBaggingID3 to SKLearn :

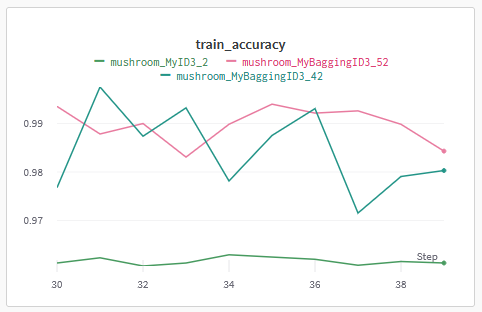
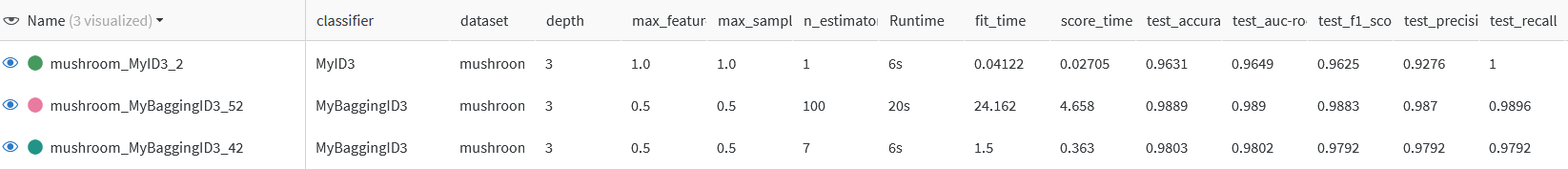
When applying the same hyper parameters on both models - we got similar performance on all datasets and all metrics, except for fit time in which SKlearn is faster.

| Same Depth, Same performance  MyID3 = SK Tree | Same performance  MyBagging = SK Bagging |
| --- | --- |
|  |  |

Example run numbers: run 20 vs run 30 ,run 29 vs run 39.

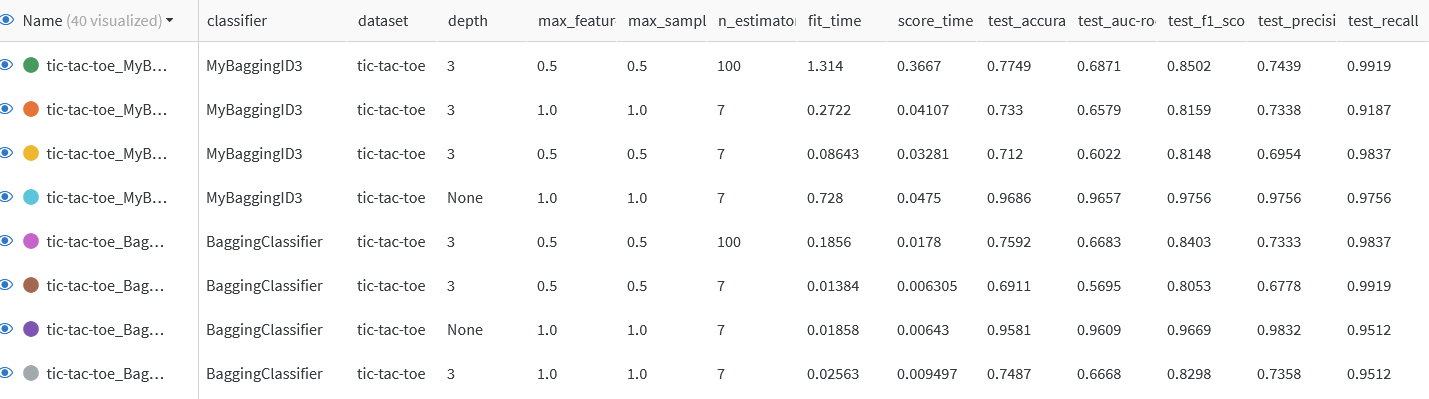
1. Impact of using a Bagging classifier with a subset of features/samples:

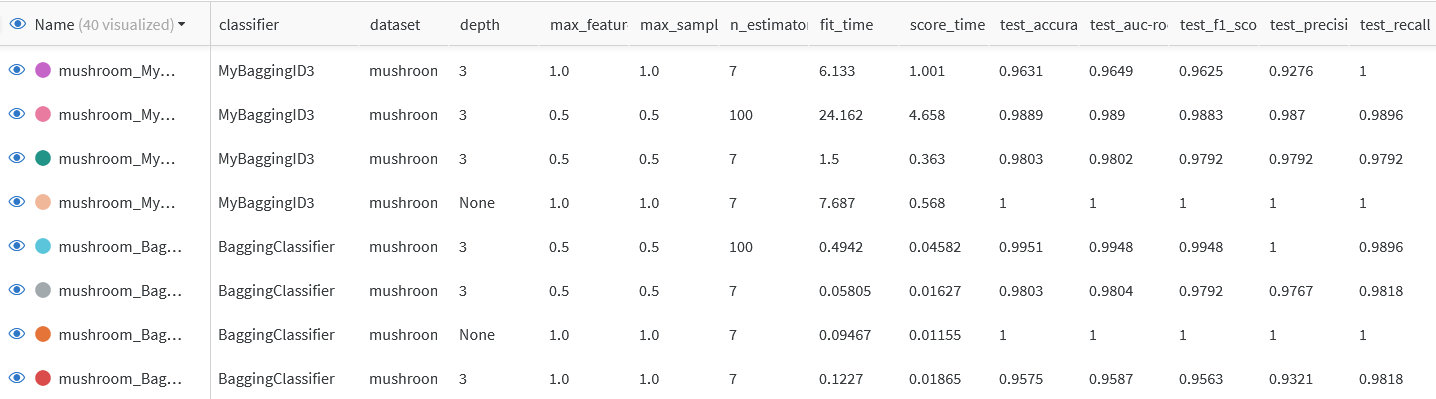
When setting max\_samples/max\_features to 0.5 in the ensembles, running many trees can still get better results than single trees, as variability is reduced, and the model becomes more resilient to over-fitting.

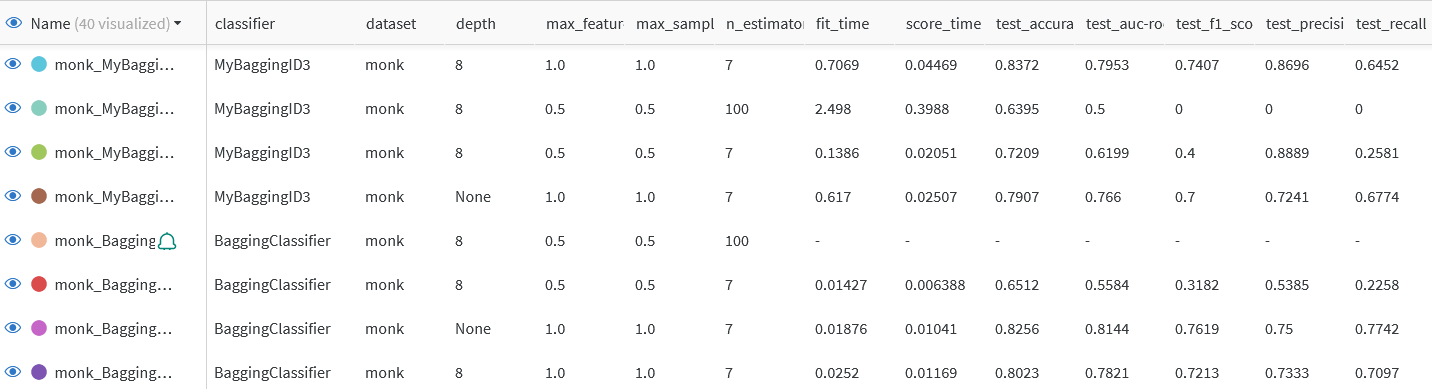
We can see that run number 52 (with n\_estimators=100 trees) outperforms both run number 42 (with 7 trees) and run number 2 (with a single tree and 100% of the features).

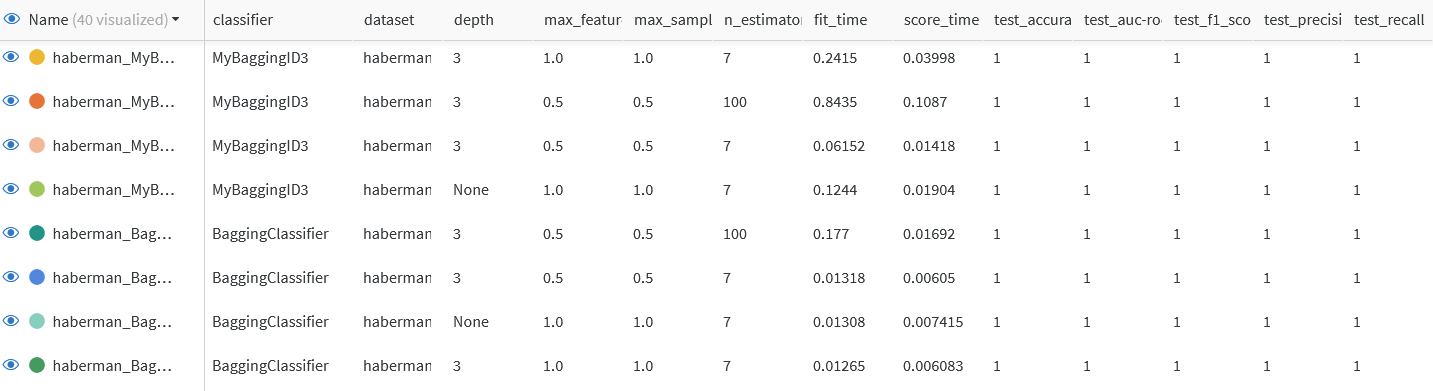
**Appendix A: configuration and results**

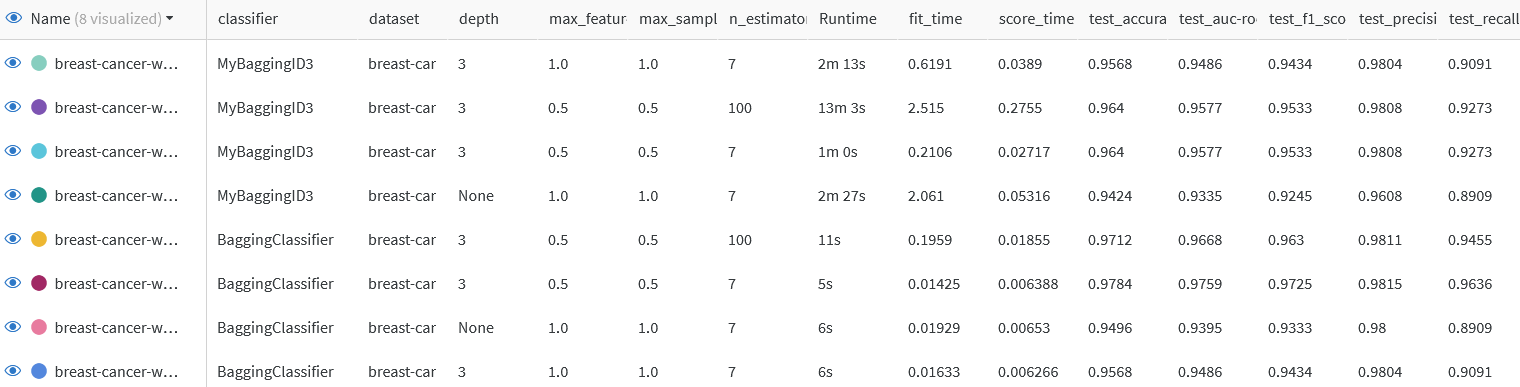
**Bagging Classifiers**

****

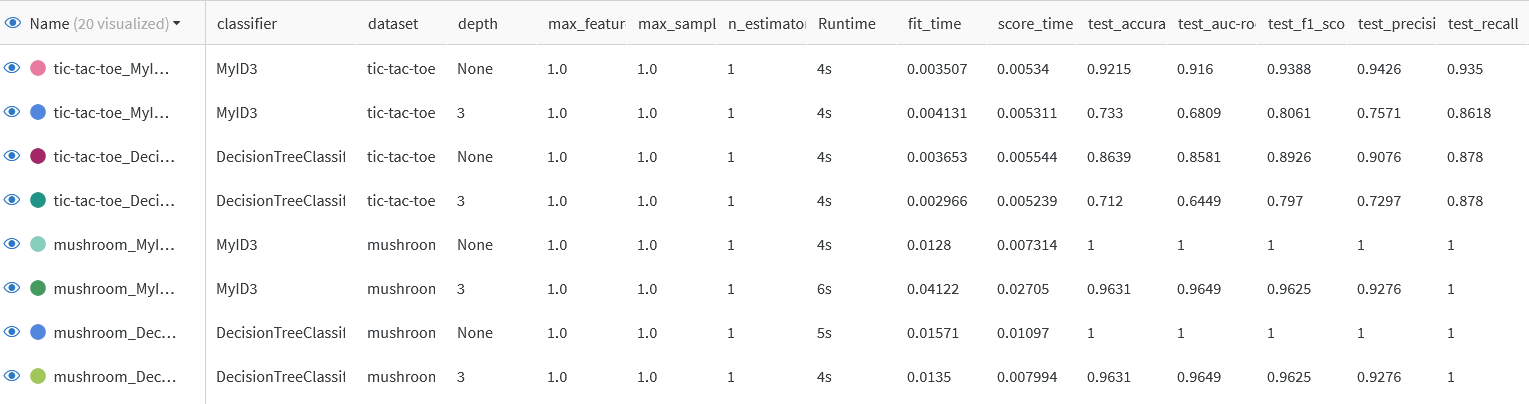
****

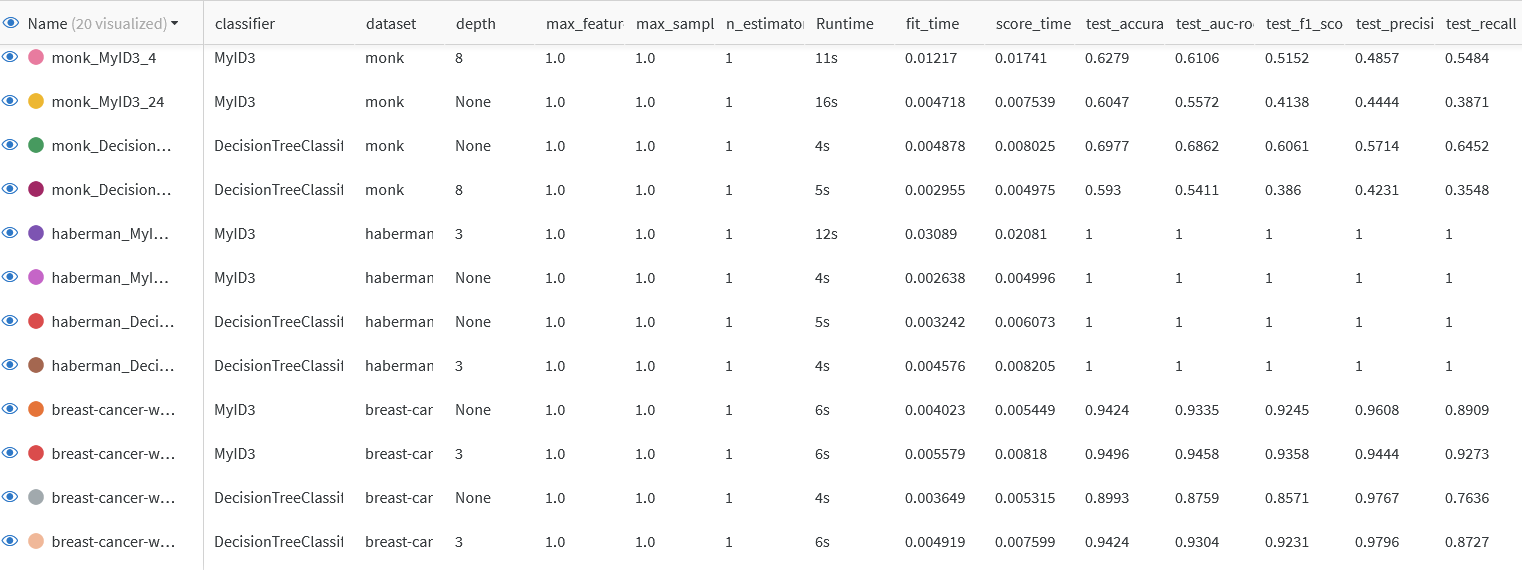
****

****

****

**Results of Single Tree Classifiers**

****

****